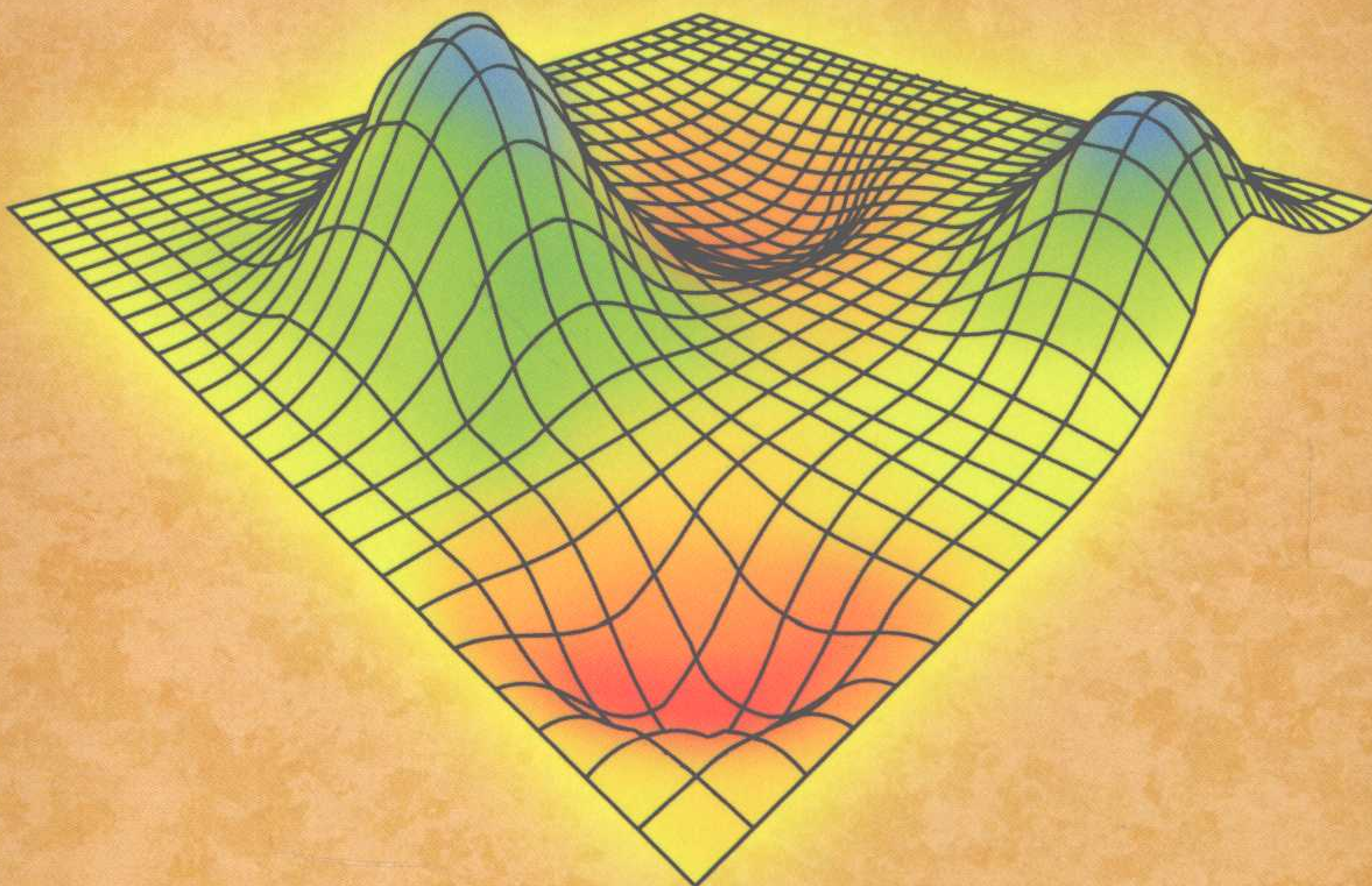
图灵数学·统计学丛书 39



Elementary Numerical Analysis

数值分析导论

（第3版）

[美] Kendall Atkinson
[中] 韩渭敏 著

王国荣 徐兆亮 孙劼 译

人民邮电出版社
POSTS & TELECOM PRESS

数值分析导论（第3版）
Elementary Numerical Analysis

本书是一本优秀的数值分析入门教材，内容全面，不仅涵盖方程的求解、插值逼近等较初级的内容，还涉及最小二乘数据拟合、本征值问题、非线性方程组等较高级主题。书中给出了大部分算法的MATLAB程序，还使用图形用户界面进行实验，学生仅需使用菜单、查询窗口和按钮就可以研究各种论题。本书习题丰富，十分有利于培养学生的独立思考和动手能力。

本书由两位著名的数值分析专家合作编写，被美国多所大学用作教材，包括加州大学伯克利分校。原书以前版本也曾有中文版问世，对国内数值分析教学产生了很大影响，被很多大学用作教材或参考书。

Kendall Atkinson 美国艾奥瓦大学荣休教授，SIAM、美国数学会、澳大利亚数学会会士。Atkinson教授著述颇丰，写有多本数值计算方面的书。此外，他还是 *Journal of Integral Equations and Applications* 的总编、*Advances in Computational Mathematics* 的编委。

韩渭敏 美国艾奥瓦大学数学系教授。1983年毕业于复旦大学数学系，1986年在中国科学院计算中心获硕士学位，1991年在美国马里兰大学获博士学位。研究范围包括计算数学、应用数学及其在工程上的应用，已发表论文上百篇，并在世界多家知名出版社出版教材和专著十余部。

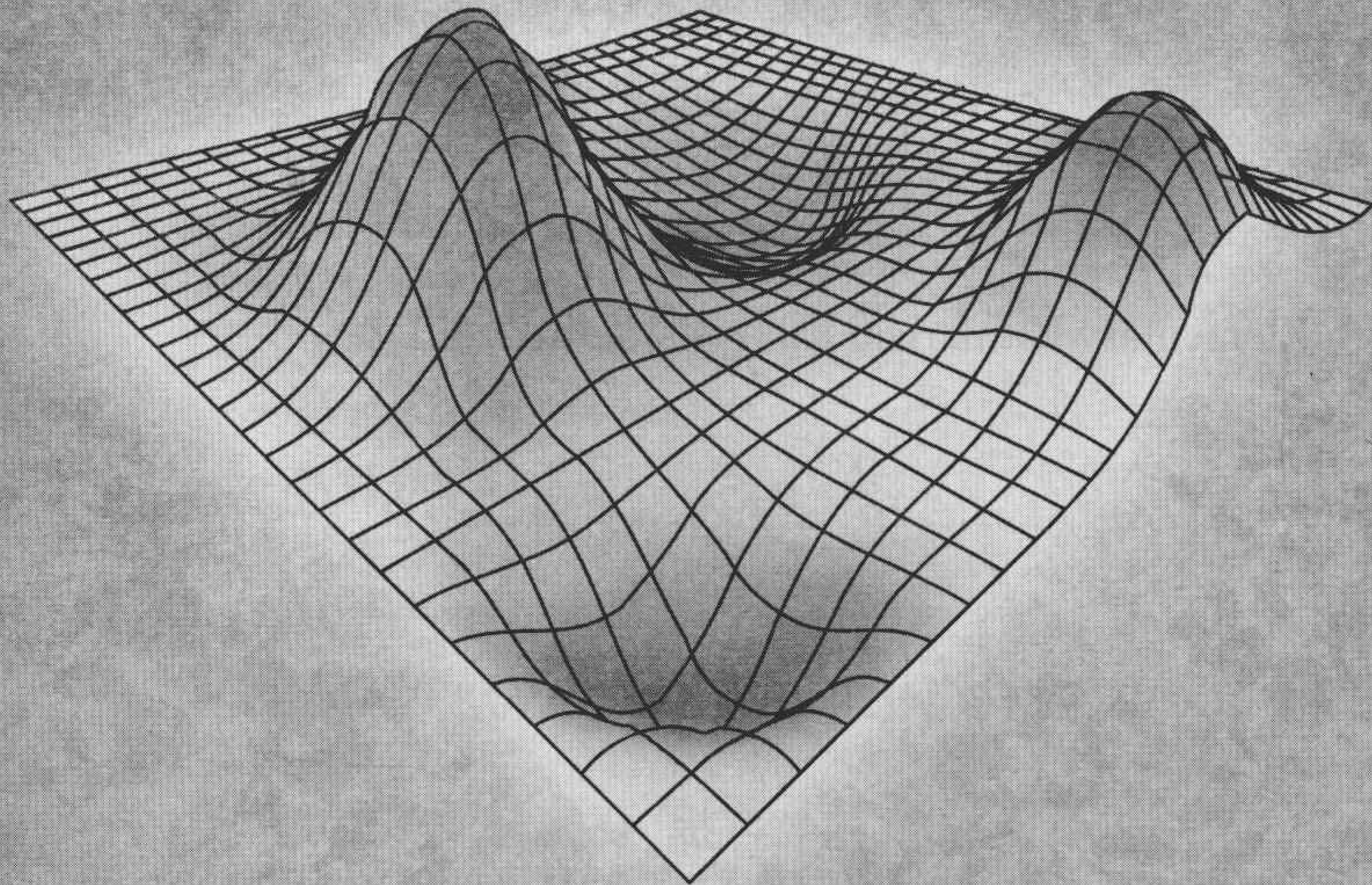
WILEY
www.wiley.com

本书相关信息请访问：图灵网站 http://www.turingbook.com
读者/作者热线：(010)51095186
反馈/投稿/推荐信箱：contact@turingbook.com
分类建议：数学/计算数学
人民邮电出版社网址 www.ptpress.com.cn

ISBN 978-7-115-21391-4
ISBN 978-7-115-21391-4
定价：69.00元

TURING 图灵数学·统计学丛书 39

WILEY



Elementary Numerical Analysis

数值分析导论

（第3版）

[美] Kendall Atkinson
[中] 韩渭敏 著

王国荣 徐兆亮 孙劼 译

人民邮电出版社
北京

图书在版编目(CIP)数据

数值分析导论：第3版/（美）阿特金森（Atkinson，K.），韩渭敏著；王国荣，徐兆亮，孙劼译. —北京：人民邮电出版社，2009. 10

（图灵数学·统计学丛书）

书名原文：Elementary Numerical Analysis, Third Edition

ISBN 978-7-115-21391-4

I. 数… II. ①阿…②韩…③王…④徐…⑤孙… III. 数值计算 IV. O 241

中国版本图书馆CIP数据核字(2009)第157625号

内容提要

本书是一本数值分析的入门教材，出自两位著名的数值分析学家，被美国多所大学用作教材. 全书包括9章，涉及方程的求解，插值与逼近，数值积分与微分，线性方程组的解等较初级的内容，以及最小二乘数据拟合、本征值问题、非线性方程组等较高级主题. 书中有大量MATLAB程序，并在附录中介绍了MATLAB. 本书习题丰富，书后还附有习题参考答案，有利于初学者自学.

本书可作为高等院校数学、工程等各理工科专业本科生的数值分析教材，也可供有关领域的研究人员和工作人员参考.

图灵数学·统计学丛书

数值分析导论（第3版）

◆ 著 [美] Kendall Atkinson [中] 韩渭敏
译 王国荣 徐兆亮 孙 劼
责任编辑 明永玲

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址：http://www.ptpress.com.cn
北京铭成印刷有限公司印刷

◆ 开本：700×1000 1/16
印张：29.5
字数：576千字 2009年10月第1版
印数：1–3 000册 2009年10月北京第1次印刷

著作权合同登记号 图字：01-2009-4229号

ISBN 978-7-115-21391-4

定价：69.00元

读者服务热线：(010)51095186 印装质量热线：(010) 67129223

反盗版热线：(010)67171154

中译本说明

本书中的程序我们用 MATLAB 语言给出. 没有 MATLAB 的读者可考虑用在互联网上能免费得到的 OCTAVE 编程语言. 书中给出的程序以基本的形式出现, 因此能容易被修改成 OCTAVE 程序.

我们热忱地感谢王国荣教授和他的学生徐兆亮教授、孙劼教授将本书译成中文. 对本书能与中国读者见面, 我们感到很高兴与荣幸.

Kendall Atkinson

韩渭敏

2009 年 8 月 25 日于艾奥瓦大学

译 者 序

美国艾奥瓦大学的 Atkinson 教授和韩渭敏教授合著的《数值分析导论 (第 3 版)》是一本优秀的入门教材. 它涵盖了数值分析课程中大部分标准课题, 不仅适合应用数学和计算数学专业的学生使用, 而且也适合工程技术专业的本科生使用.

本书的目的是使学生对数值分析的一些数值计算方法以及计算过程中的误差概念和理论有初步的了解. 书中给出了大部分算法的 MATLAB 程序, 可避免编写过多的程序, 大大地节省了学生的时间. 同时, 指导学生从网上得到有关的程序及其他的课程材料, 鼓励学生实验性地学习数值方法.

作为一本教材, 它配有许多习题. 有些习题可作为有关理论的进一步说明或深入研究的材料. 其中大多数习题能用计算器或计算机程序完成, 十分利于培养学生的独立思考和动手能力.

本书作者 Atkinson 教授是我的老朋友, 1988~1989 年我作为访问学者在艾奥瓦大学数学系聆听过他的精彩讲课并参加他主持的讨论班, 得到他的许多帮助. 韩渭敏教授是我的新朋友, 当他得知我要翻译本书时, 主动热情地帮助审读译稿, 在内容和文字上提出了许多宝贵的改进意见, 为本书翻译质量的提高作出了重要的贡献. 在此, 我要向他们表示衷心的感谢.

本书的翻译工作是由我和我以前的研究生 —— 上海海事大学徐兆亮教授和上海应用技术学院孙劼教授合作完成的. 其中第 3~5 章由徐兆亮翻译, 第 6 章和第 7 章由孙劼翻译, 我负责翻译其余部分并统稿.

本书是应人民邮电出版社图灵公司之邀翻译的, 翻译时间比较紧迫, 译文中难免有不当之处, 欢迎广大读者批评指正.

王国荣

2009 年于上海师范大学

前　　言

本书是一本数值分析的入门教材，适合数学和其他理工科专业的大学生使用. 学生应具备的主要的基础知识是一元函数微积分的一学年课程所讲的内容，也要求对计算机有些了解. 本书可用于大学本科生的数值分析课程. 本书最后四章介绍线性代数、常微分方程和偏微分方程的数值方法. 对这些题材的背景知识有所了解是很有用的，当然最后四章也有对这些题材理论方面必要的介绍.

学生选读数值分析课程有各种各样的原因. 某些人在学习其他科目、从事研究工作或者在他们的专业中需要数值分析，另一些人学习它则是为了扩充其科学计算知识. 我们讲授这门课时，给学生设置了几个目标. 首先，他们将对解决数值分析基本问题 (如各章标题所述) 的一些数值方法有初步的了解并能运用；其次，他们将了解误差概念并懂得为什么需要分析和预估误差；第三，他们将逐步积累用计算机实现数值方法的某些经验，这些经验应该包括理解计算机算术运算及其结果.

本书包含数值分析课程中大部分标准主题，而且也探究了本学科的一些重要的基础论题. 其中包括复杂问题用比较简单的问题逼近、算法的构造、迭代法、误差分析、稳定性、渐近误差公式以及机器算术运算的结果. 考虑到课程的层次，重点放在对于问题本身和用于求解这些问题的数值方法的直观理解上. 我们精心选择了例子来帮助加深这种理解，而不仅仅为了说明算法. 我们只在证明足够简单和对结果的直观理解有所帮助时才会给出证明.

关于计算机程序设计，数值分析入门课程中首选的语言是 MATLAB. 附录 D 简要介绍了 MATLAB；而教材中的程序是更深入的例子. 我们鼓励学生修改这些程序并把它们作为编写自己的 MATLAB 程序的模型. 作者讲授这门课程时，还提供一些在线 MATLAB 学习材料的网站链接.

在书中包含MATLAB程序有多种原因. 首先，它们用于说明算法的构造. 其次，可以节省学生的时间，避免他们编写过多的程序，从而有更多时间实践这些数值方法. 毕竟，课程的重心应该是数值分析而不是学习如何编程. 第三，这些程序提供MATLAB语言的例子以及使用MATLAB时一些比较好的程序设计实践的例子. 当然，学生们应该自己编写一些程序. 有些程序可以通过简单修改教材中的一些程序而得到，例如，修改梯形求积法的程序可以得到中点法的程序. 而其他程序则是需要较多原创的. 本书中的所有程序可以从 John Wiley 出版社的网站 `http://www.wiley.com/college/atkinson` 上得到. 我们也提供一个站点`http://www.math.uiowa.edu/ftp/atkinson/ENA_Materials`，其中包括这些程序和其他课

程材料.

除了在书中使用 MATLAB 程序之外, 我们还使用图形用户界面 (GUI) 进行实验, 让学生仅需使用菜单、查询窗口和按钮就能研究各种论题. 我们已经编写出了其中的好几个, 包括用于构造和分析泰勒多项式逼近, 求根, 既用等距节点又用切比雪夫节点进行多项式插值, 以及数值积分. GUI 是使用 MATLAB GUI 开发环境编写的, 它们必须在 MATLAB 中运行. 这些 GUI 从上面给出的本书作者网站可以得到, 欢迎教师、学生以及使用 GUI 的其他人给我们发来反馈信息.

本书每一节末尾都有一些习题. 习题分为几种类型. 有些习题进一步说明本节中给出的理论结果, 其中的大多数能用计算器或简单的计算机程序完成. 其他是为了进一步探究本节的理论性材料, 或者拓展其他理论结果. 在某些小节中给出的习题需要相当大的程序量.

与第 2 版相比, 第 3 版新增了一章和两小节, 并重新组织了内容. 重写了计算机算术运算一节, 将重点放在计算机中表示数的 IEEE 浮点形式. 二进制算术运算一节被移到了新的附录 E 中. 新的两节是关于函数的最小二乘逼近的 4.7 节 [包含介绍勒让德 (Legendre) 多项式] 以及关于两点边值问题的 8.8 节. 新增的第 9 章讨论了经典的两个变量的二阶线性偏微分方程的数值方法. 此外, 我们改写了教材的其他若干部分, 并且增加了例题和许多习题.

用本教材讲授一学期的课程时, 我们通常讲授第 1~6 章和第 8 章的大部分内容, 第 6 章的线性代数知识可以随时讲, 但我们通常留在课程的后半段讲授. 第 4 章中关于多项式插值的知识需要在第 5 章和第 8 章之前讲授. 对一学期的课程而言, 本书含有足够多的内容, 教师在考虑略去哪些内容时有相当大的回旋余地.

感谢艾奥瓦大学的同事们对教材提出的意见, 谢谢 Paul Lorczak 检查了手稿的准确性. 我们要感谢下列审稿人的建议, 他们的意见对于第 3 版的写作十分有益: Richard Braun, Prabir Daripa, David Horntrop, Barbara Shabell, John Strain, Jingyi Zhu. 还要感谢 Cymie Wehr 为第 2 版准备的 LATEX 排版, 非常之棒. 我们在此基础上写第 3 版, 节省了大量时间和精力. 谢谢 Brian Treadway 教我们熟悉 LATEX, 他的帮助是无价的. John Wiley 出版公司的员工们始终为本书而努力着, 我们也要感谢他们.

Kendall Atkinson

Weimin Han(韩渭敏)

2003 年 5 月于艾奥瓦州艾奥瓦城

目　录

第 1 章　泰勒多项式

数值分析使用的结论与方法来自数学的许多领域, 特别是微积分和线性代数. 本章介绍一个来自微积分的非常有用的工具 —— 泰勒定理. 这个工具对本书中讨论的许多数值方法的形成和理解是十分必要的.

1.1 节引入泰勒多项式作为其他函数近似求值的一个方法; 1.2 节给出求泰勒多项式逼近误差的精确公式 —— 泰勒定理. 最后在 1.3 节中, 我们首先讨论如何求多项式的值, 然后以一个具体的函数为例推导和分析一个可计算的多项式逼近.

代数和微积分的其他有关知识在附录中给出. 附录 A 复习中值定理, 附录 B 复习微积分、代数、几何、三角函数的其他一些内容.

我们可以用多种计算机语言编写程序, 实施本教材中学到的数值方法. 最重要的基本计算机语言是 C、C++、Java 以及 Fortran. 本教材中使用一种高级语言, 用它在求解数学问题实施数值分析过程中更容易处理我们需要的数学结构. 这种语言是 MATLAB, 它广泛使用在各种类型的计算机上. 本教材提供许多 MATLAB 程序的例子, 我们鼓励学生使用这些程序, 并修改它们解决类似的任务. 附录 D 非常简要地介绍 MATLAB, 并列出一些更详尽地介绍 MATLAB 的文献.

1.1　泰勒多项式

在数学中出现的大多数函数 $f(x)$ 都不能用简单的方法精确地求值. 例如, 如果不用计算器或计算机求 $f(x) = \cos x$, e^x 或 $\sqrt{x}$ 的值, 我们就得使用几乎等于 $f(x)$ 并且较容易求值的函数 $\hat{f}(x)$. 最常用的一类近似函数 $\hat{f}(x)$ 是多项式. 它们容易处理并且通常是逼近 $f(x)$ 的有效工具. 另一种相关的函数形式是分段多项式函数, 它也被广泛地使用, 我们将在 4.3 节中研究它.

在多项式中, 使用最广泛的是泰勒多项式. 对具体的问题, 往往有更有效的逼近多项式, 第 4 章将讨论一些这样的多项式. 泰勒多项式相当容易构造, 并且它通常是得到更有效逼近的第一步. 泰勒多项式在多个数学领域中都很重要.

设 $f(x)$ 表示一个给定的函数, 例如 e^x, $\sin x$ 或 $\ln(x)$. 假定 $f(x)$ 在 $x = a$ 及其附近有定义, 则在 $x = a$ 处的泰勒多项式模拟 $f(x)$ 在点 $x = a$ 的性态, 并在接近于 a 的点 x 上几乎等于 $f(x)$.

举一个具体的例子, 求线性多项式 $p_1(x)$, 使其满足

$$\begin{aligned} p_1(a) &= f(a) \\ p_1'(a) &= f'(a) \end{aligned} \tag{1.1}$$

容易验证, 多项式由

$$p_1(x) = f(a) + (x-a)f'(a) \tag{1.2}$$

唯一确定. $y = p_1(x)$ 的图像是 $y = f(x)$ 在 $x = a$ 处的切线.

例 1.1.1 设 $f(x) = \mathrm{e}^x$ 且 $a = 0$, 则

$$p_1(x) = 1 + x$$

f 和 p_1 的图像在图 1-1 中给出. 注意, 当 x 接近于 0 时, e^x 近似地为 $p_1(x)$. ■

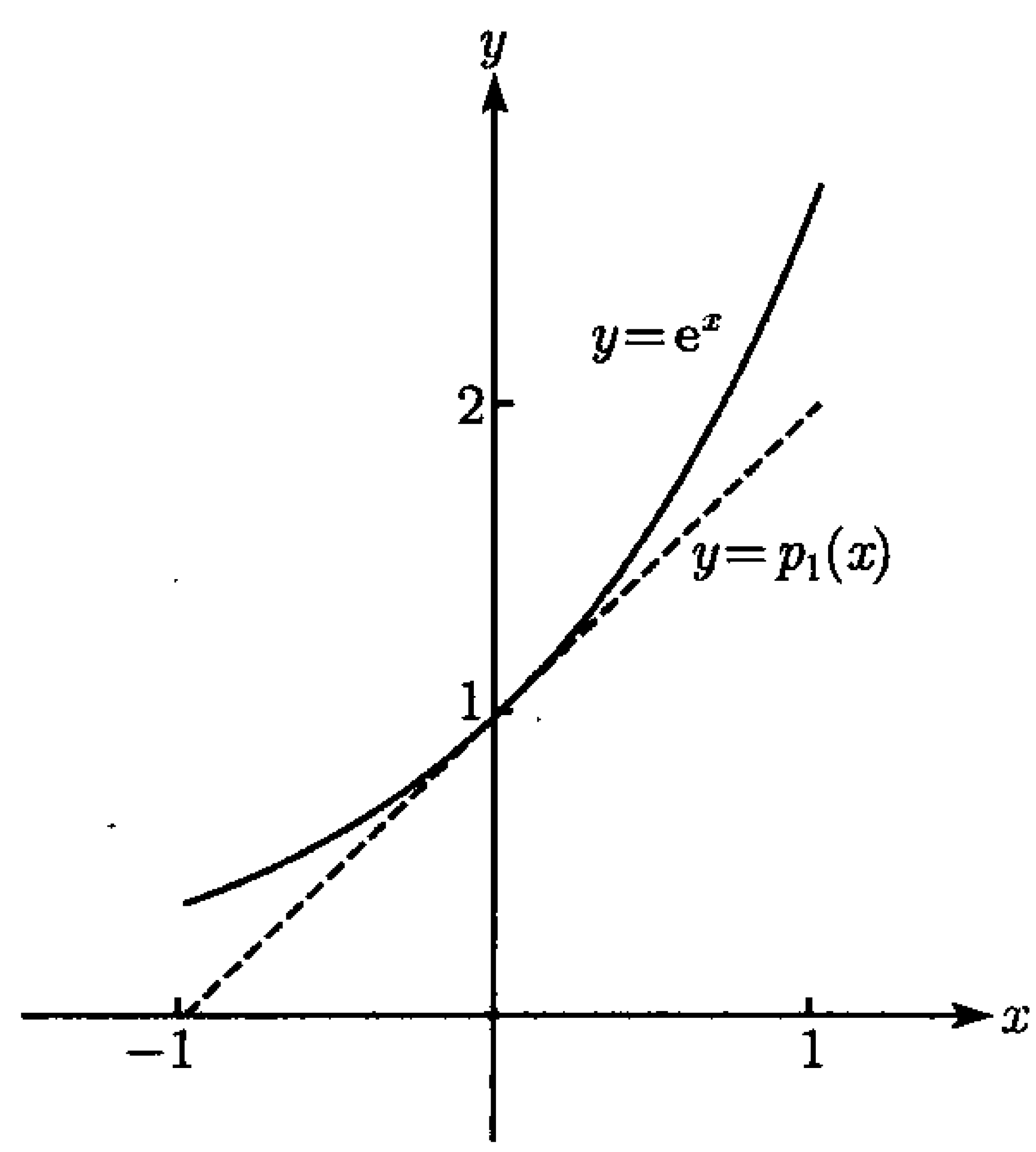


图 1-1 线性泰勒逼近

继续这样的构造过程, 考虑求一个在 $x = a$ 附近逼近 $f(x)$ 的二次多项式 $p_2(x)$. 因为在二次多项式公式中有三个系数, 比如说,

$$p_2(x) = b_0 + b_1 x + b_2 x^2$$

自然需要利用关于 $p_2(x)$ 的三个条件决定它们. 为较好地模仿 $f(x)$ 在 $x = a$ 处的性态, 要求

$$\begin{aligned} p_2(a) &= f(a) \\ p_2'(a) &= f'(a) \\ f_2''(a) &= f''(a) \end{aligned} \tag{1.3}$$

可以检验, 公式

$$p_2(x) = f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a) \tag{1.4}$$

满足这些条件.

例 1.1.2 继续前面的例子, 考虑 $f(x)=\mathrm{e}^x$ 和 $a=0$, 我们有

$$p_2(x)=1+x+\frac{1}{2}x^2$$

函数 e^x, $p_1(x)$ 和 $p_2(x)$ 的对比可见图 1-2. ■

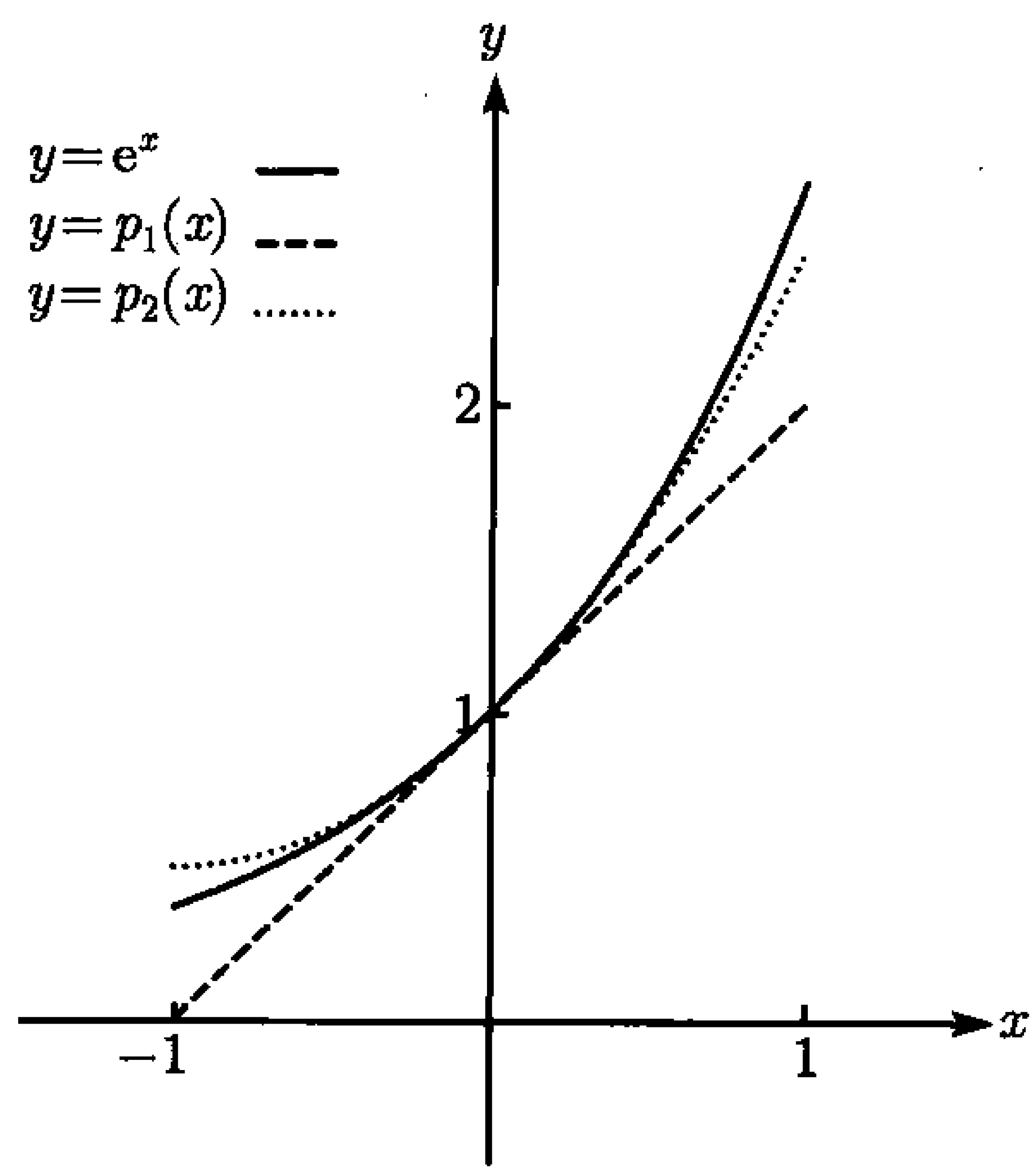


图 1-2 线性逼近和二次泰勒逼近

我们可以继续模拟 $f(x)$ 在 $x=a$ 处的性态. 设 $p_n(x)$ 是一个 n 次多项式, 满足

$$p_n^{(j)}(a)=f^{(j)}(a),\quad j=0,1,\cdots,n \tag{1.5}$$

其中 $f^{(j)}(x)$ 是 $f(x)$ 的 j 阶导数. 可以验证

$$\begin{aligned}p_n(x)&=f(a)+(x-a)f'(a)+\frac{(x-a)^2}{2!}f''(a)+\cdots+\frac{(x-a)^n}{n!}f^{(n)}(a)\\&=\sum_{j=0}^{n}\frac{(x-a)^j}{j!}f^{(j)}(a)\end{aligned} \tag{1.6}$$

在公式中, $f^{(0)}(a)=f(a)$, 而且

$$j!=\begin{cases}1, & j=0\\ j(j-1)\cdots(2)(1), & j=1,2,3,4,\cdots\end{cases}$$

称为 "j 阶乘". 如果在 (1.6) 中需要明确地表明与点 a 有关, 就把它记作 $p_n(x;a)$. (1.6) 中的多项式 $p_n(x)$ 称为函数 $f(x)$ 在近似点 a 处的 *n次的泰勒多项式*. [注: 若 $f^{(n)}(a)=0$, 则多项式 $p_n(x)$ 的实际次数小于 n.]

例 1.1.3 仍考虑 $f(x)=\mathrm{e}^x$ 和 $a=0$. 则

$$f^{(j)}(x)=\mathrm{e}^x,\quad f^{(j)}(0)=1,\quad \text{对一切}\quad j\geqslant 0$$

于是

$$p_n(x)=1+x+\frac{1}{2!}x^2+\cdots+\frac{1}{n!}x^n=\sum_{j=0}^{n}\frac{x^j}{j!} \tag{1.7}$$

表 1-1 包括 $[-1, 1]$ 中 x 的几个典型值上 $p_1(x)$, $p_2(x)$, $p_3(x)$ 以及 e^x 的值. 对固定的 x, 当次数 n 增加时精度提高. 对固定次数的多项式, 当 x 远离 $a=0$ 时准确度降低. ■

表 1-1 e^x 的泰勒逼近

x	$p_1(x)$	$p_2(x)$	$p_3(x)$	e^x
−1.0	0	0.500	0.333 33	0.367 88
−0.5	0.5	0.625	0.604 17	0.606 53
−0.1	0.9	0.905	0.904 83	0.904 84
0	1.0	1.000	1.000 00	1.000 00
0.1	1.1	1.105	1.105 17	1.105 17
0.5	1.5	1.625	1.645 83	1.648 72
1.0	2.0	2.500	2.666 67	2.718 28

例 1.1.4 设 $f(x)=\mathrm{e}^x$, 且设点 a 是一个任意点, 不一定是 0. 则

$$f^{(j)}(x)=\mathrm{e}^x,\quad f^{(j)}(a)=\mathrm{e}^a,\quad \text{对一切}\quad j\geqslant 0$$

因此

$$p_n(x;a)=\mathrm{e}^a\left[1+(x-a)+\frac{1}{2!}(x-a)^2+\cdots+\frac{1}{n!}(x-a)^n\right]=\mathrm{e}^a\sum_{j=0}^{n}\frac{(x-a)^j}{j!}$$

例如,

$$p_n(x;1)=\mathrm{e}^1\sum_{j=0}^{n}\frac{(x-1)^j}{j!} \tag{1.8}$$

多项式 $p_n(x;1)$ 当 $x\approx 1$ 时最准确; (1.7) 给出的多项式 $p_n(x;0)$ 当 $x\approx 0$ 时最准确. 作为一个习题, 要求读者对不同的 n 值, 在区间 $[-1,2]$ 上比较 $p_n(x;0)$ 和 $p_n(x;1)$ 的准确度 (参见习题 8). ■

注记 本书中使用两个符号表示 "近似地等于". 符号 "≈" 通常用于近似关系, 例如

$$x\approx 5$$

表示 x 在 5 的附近; 而

$$\mathrm{e}^x\approx 1+x,\quad x\approx 0$$

表示当 x 在 0 的附近时, e^x 近似地为 $1+x$. 符号 "≐" 通常用于数, 如

$$2\pi \doteq 6.283\,2$$

$$\sqrt{168} \doteq 12.961$$

符号 "≐" 通常用于实际的计算误差, 我们试图在用法上前后一致, 但有时并不清楚使用哪一个符号更恰当. ■

例 1.1.5 设 $f(x)=\ln(x)$ 且 $a=1$. 则 $f(1)=\ln(1)=0$, 由归纳法, 对 $j\geqslant 1$,

$$f^{(j)}(x)=(-1)^{j-1}(j-1)!\frac{1}{x^j}$$

$$f^{(j)}(1)=(-1)^{j-1}(j-1)!$$

利用公式 (1.6), 我们得到下列泰勒多项式

$$\begin{aligned} p_n(x) &= (x-1)-\frac{1}{2}(x-1)^2+\frac{1}{3}(x-1)^3-\cdots+(-1)^{n-1}\frac{1}{n}(x-1)^n \\ &= \sum_{j=1}^{n}\frac{(-1)^{j-1}}{j}(x-1)^j \end{aligned} \tag{1.9}$$

$\ln(x)$, $p_1(x)$, $p_2(x)$ 以及 $p_3(x)$ 的图像见图 1-3. ■

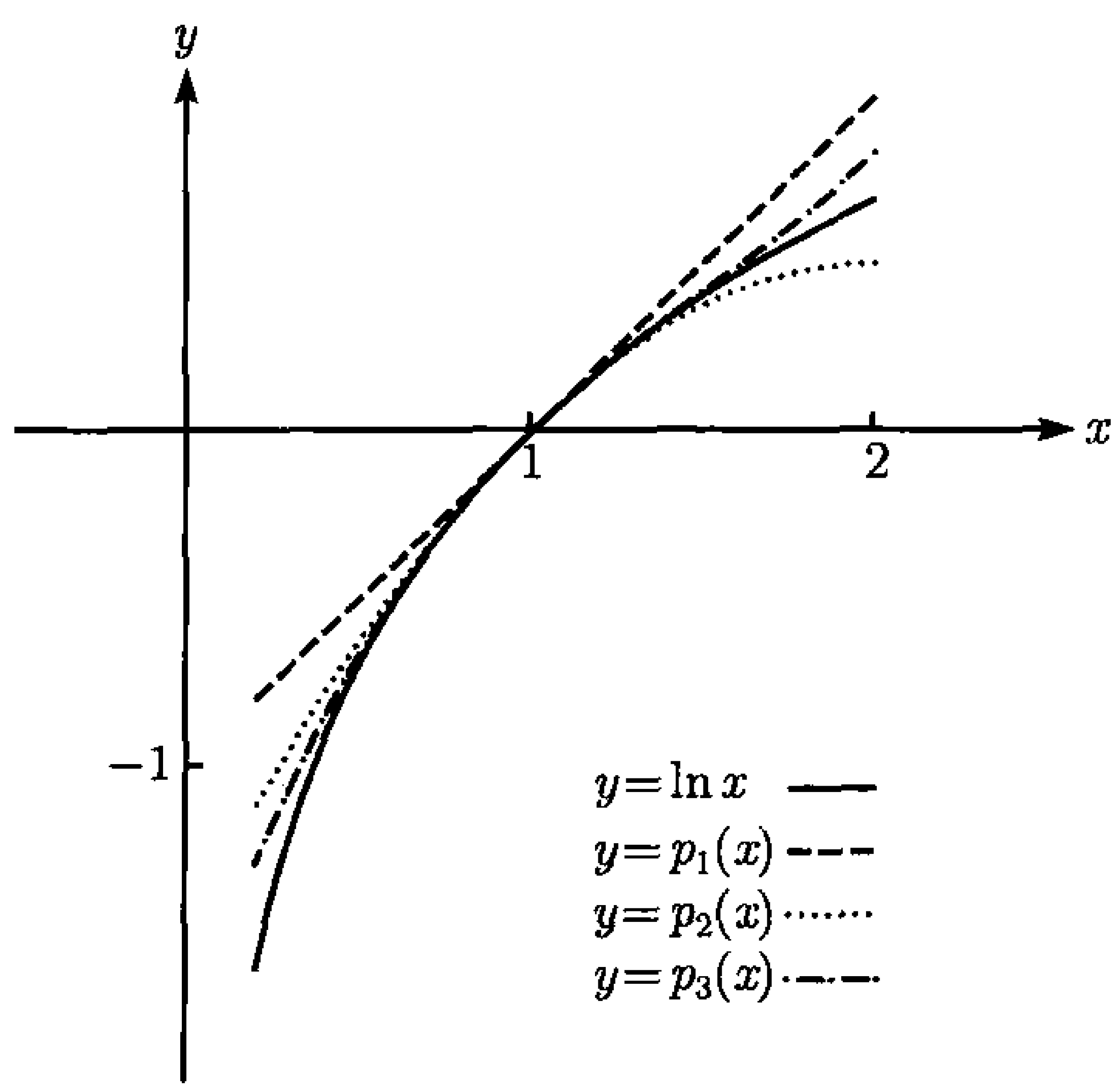


图 1-3 $\ln(x)$ 关于 $x=1$ 的泰勒逼近

本书中, 我们将在适当的地方给出一些数值分析中的一般准则 (general observation).

> **一般准则**
>
> 当不知道求解一个数学问题的直接方法时, 我们就用一个能够计算解的 "相近的问题 (nearby prolem)" 替代它. (1.10)

在本节中, 我们就用多项式求值替代诸如 e^x 这样的函数求值.

MATLAB 程序: 泰勒多项式求值 下面是一个 MATLAB 程序, 它在区间 $[-b, b]$ 上计算若干个泰勒多项式逼近 e^x, 其中 b 的值输入到程序中. 在区间 $[-b, b]$ 内选择的点 x 上, 程序将计算 1 次、2 次、3 次以及 4 次泰勒多项式的值. 以列表形式输出它们各自的计算误差. 既在用户的计算机屏幕上显示又在文件exp_taylor中保留.

```
% TITLE: Evaluate Taylor polynomials for exp(x) about x = 0
%
% This evaluates several Taylor polynomials and their errors
% for increasing degrees.  The particular function being
% approximated is exp(x) on [-b,b].

% Initialize
b = input('Give the number b defining the interval [-b,b] ');
h = b/10;
x = -b:h:b;
max_deg = 4;

% Produce the Taylor coefficients for the function exp(x) when
% expanded about the point a = 0.  The coefficients are stored
% in the array c, which will have length max_deg+1.

c = ones(max_deg+1,1);
fact = 1;
for i = 1:max_deg
  fact = i*fact;
  c(i+1) = 1/fact;
end

% Calculate the Taylor polynomials
p1 = polyeval(x,0,c,1);
p2 = polyeval(x,0,c,2);
p3 = polyeval(x,0,c,3);
p4 = polyeval(x,0,c,4);

% Calculate the errors in the Taylor polynomials
true = exp(x);
err1 = true-p1;
err2 = true-p2;
err3 = true-p3;
err4 = true-p4;

% Print the errors in tabular format
diary exp_taylor
disp('    x        exp(x)      err1       err2       err3       err4')
```

```
for i = 1:length(x)
  fprintf('%7.3f%10.3f%14.3e%14.3e%14.3e%14.3e\n',...
          x(i),true(i),err1(i),err2(i),err3(i),err4(i))
end
diary off
```

上述程序使用下列取名为polyeval的计算多项式的程序. 程序中使用的方法将在1.3 节中讨论.

```
function value = polyeval(x,alpha,coeff,n);
%
% function value = polyeval(x,alpha,coeff,n)
%
% Evaluate a Taylor polynomial at the points given in x, with
% alpha the point of expansion of the Taylor polynomial, and
% with n the degree of the polynomial.  The coefficients are to
% be given in coeff; and it is assumed there are n+1 entries in
% coeff with coeff(1) the constant term in the polynomial

value = coeff(n+1)*ones(size(x));
z = x-alpha;
for i = n:-1:1
  value = coeff(i) + z.*value;
end
```

习题

1. 利用(1.9), 按表 1-1 的方式比较 $\ln(x)$ 及其 1 次、2 次以及 3 次泰勒多项式, 要求在区间 $[\frac{1}{2}, \frac{3}{2}]$ 中进行.
2. 对下列情形导出线性泰勒多项式和二次泰勒多项式. 画出给定的函数以及它们的泰勒多项式的图像.
   (a) $f(x) = \sqrt{x}, a = 1$　　(b) $f(x) = \sin(x), a = \frac{\pi}{4}$
   (c) $f(x) = e^{\cos(x)}, a = 0$　　(d) $f(x) = \ln(1 + e^x), a = 0$
3. 对下列函数, 利用 $a = 0$ 作为逼近点, 导出 n 次泰勒多项式的一般公式.
   (a) $1/(1-x)$　　(b) $\sin(x)$　　(c) $\sqrt{1+x}$
   (d) $\cos(x)$　　(e) $(1+x)^{\frac{1}{3}}$
4. 函数 $f(x) = \sqrt[3]{x}$ 在 $x = 0$ 处有 1 次泰勒多项式逼近吗? 在 $x = 1$ 处呢? 解释并验证你的答案.
5. 利用习题 3c 中得到的函数 $f(x) = \sqrt{1+x}$ 的 1 次、2 次和 3 次泰勒多项式, 计算 $\sqrt{0.9}, \sqrt{1.1}, \sqrt{1.5}, \sqrt{2.0}$ 的近似值, 并且把它们与 8 位或多于 8 位的精确值比较.
6. 重复习题 5, 但这次 $f(x) = \sin(x)$, $x = 0.01, 0.1, 0.5, 1.0$.
7. 在区间 $[-\frac{\pi}{2}, \frac{\pi}{2}]$ 上, $a = 0$ 处, 比较 $f(x) = \sin(x)$ 与它的 1 次、3 次和 5 次泰勒多项式. 以表 1.1 的方式创建一个表格.
8. 设 $f(x) = e^x$; 分别回想 $p_n(x;0)$ 和 $p_n(x;1)$ 的公式 (1.7) 和 (1.8). 在区间 $[-1,2]$ 上, 对

n =1, 2, 3, 把 $p_n(x;0)$ 和 $p_n(x;1)$ 分别与 e^x 作比较. 讨论结果.

9. **(a)** 对 $f(x)=e^{-x}$, 用 $a=0$ 作为逼近点, 导出 1 次、2 次、3 次以及 4 次泰勒多项式.

(b) 利用 e^t 的泰勒多项式, 代入 $t=-x$ 得到 e^{-x} 的多项式逼近. 把这个结果与 (a) 中的结果作比较.

10. **(a)** 对 $f(x)=e^{x^2}$, 用 $a=0$ 作为逼近点, 导出 1 次、2 次、3 次以及 4 次泰勒多项式.

(b) 利用 e^t 的泰勒多项式, 代入 $t=x^2$ 得到 e^{x^2} 的多项式逼近. 把这个结果与 (a) 中的结果作比较.

11. 商

$$g(x)=\frac{e^x-1}{x}$$

当 $x=0$ 时无定义. 依次利用函数 e^x 的 1 次、2 次和 3 次泰勒多项式逼近来确定 $g(0)$ 的一个自然的定义.

12. 当 $x=0$ 时, 商

$$g(x)=\frac{\ln(1+x)}{x}$$

无定义. 依次利用函数 $\ln(1+x)$ 的 1 次、2 次和 3 次泰勒多项式逼近来确定 $g(0)$ 的一个自然的定义.

13. **(a)** 作为线性泰勒多项式的替代, 现在构造线性多项式 $q(x)$, 使其满足: 对给定的点 a 和 b,

$$q(a)=f(a), \quad q(b)=f(b).$$

(b) 设 $f(x)=e^{x^2}$, $a=0$, $b=1$, 对 $0\leqslant x\leqslant 1$, 从数值上比较由 (a) 构造的 $q(x)$ 和线性泰勒多项式.

14. 对 $f(x)=e^{x^2}$, 构造三次多项式 $q(x)$, 使其满足

$$q(0)=f(0), \quad q(1)=f(1)$$

$$q'(0)=f'(0), \quad q'(1)=f'(1)$$

对 $0\leqslant x\leqslant 1$, 把 $q(x)$ 分别与 e^x 和 (1.6) 给出的泰勒多项式 $p_3(x)$ 作数值上的比较.
提示: 记 $q(x)=b_0+b_1x+b_2x^2+b_3x^3$. 从 $x=0$ 的条件确定 b_0 和 b_1. 然后得到关于系数 b_2 和 b_3 的两个方程的线性方程组.

1.2 泰勒多项式的逼近误差

实际使用泰勒多项式逼近 $f(x)$ 时, 我们必须知道它的精度. 下述定理是估计精度的主要工具. 因为在大多数微积分教材中都讨论这个定理, 所以, 我们只描述它而不加证明.

定理 1.2.1(泰勒余项) 假设在区间 $\alpha\leqslant x\leqslant\beta$ 上, $f(x)$ 有 $n+1$ 阶连续导数, 并设点 a 属于该区间. 对 (1.6) 的泰勒多项式 $p_n(x)$, 用 $R_n\equiv f(x)-p_n(x)$ 表

示 $p_n(x)$ 逼近 $f(x)$ 的余项或误差. 则

$$R_n(x) = \frac{(x-a)^{n+1}}{(n+1)!} f^{(n+1)}(c_x), \quad \alpha \leqslant x \leqslant \beta \tag{1.11}$$

其中 c_x 是 a 和 x 之间的一个未知点.

例 1.2.2 设 $f(x) = \mathrm{e}^x$, $a = 0$. 相应的泰勒多项式由 (1.7) 给出. 从上面的定理知道, 逼近误差为

$$\mathrm{e}^x - p_n(x) = \frac{x^{n+1}}{(n+1)!}\mathrm{e}^c, \quad n \geqslant 0 \tag{1.12}$$

其中 c 在 0 和 x 之间. 利用这个公式可以证明, 对每个固定的 x, 当 $n \to \infty$ 时误差趋向于零; 当 $|x| \leqslant 1$ 时, 这个结论很显然. 由这个公式还知, 对每个固定的 n, 当 x 离开 0 时, 误差变大. 为了从图形上说明这点, 和 1.1 节的做法不同, 这里考虑误差 $\mathrm{e}^x - p_n(x)$ 的图形, 而不直接用图形表示函数 e^x 和多项式 $p_n(x)$. 图 1-4 画出区间 $[-1,1]$ 上次数为 $n = 1,2,3,4$ 的泰勒多项式的误差. 从图上可以看到所述的关于泰勒多项式的逼近误差的性态.

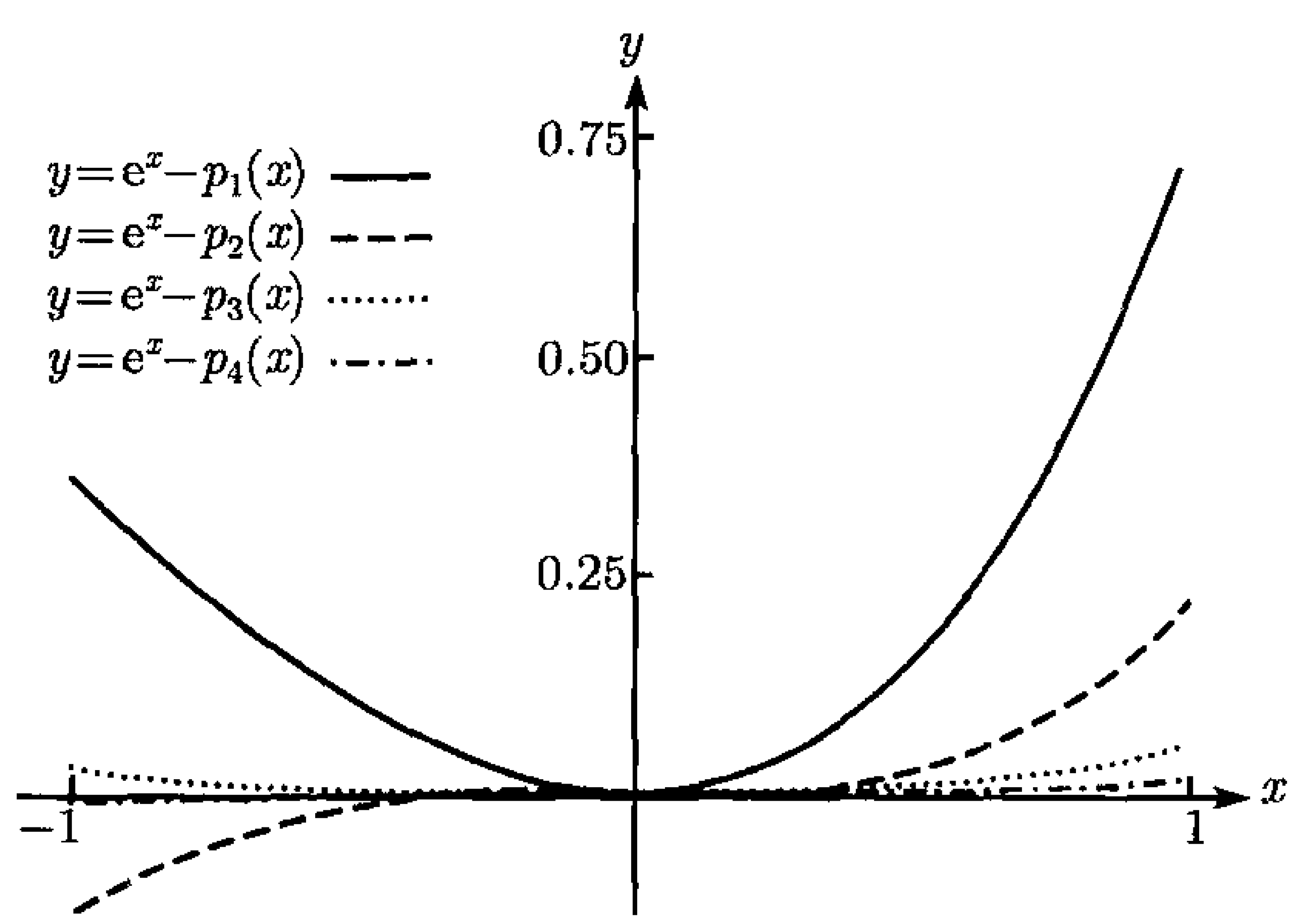


图 1-4 泰勒多项式逼近 e^x 中的误差

例 1.2.3 作为 (1.12) 的特例, 设 $x = 1$. 则从 (1.7) 可得

$$\mathrm{e} \approx p_n(1) = 1 + 1 + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

从 (1.12) 知误差为

$$\mathrm{e} - p_n(1) = R_n(1) = \frac{\mathrm{e}^c}{(n+1)!}, \qquad 0 < c < 1$$

因为 $\mathrm{e} < 3$, 所以可界定 $R_n(1)$ 如下

$$\frac{1}{(n+1)!} \leqslant R_n(1) \leqslant \frac{\mathrm{e}}{(n+1)!} < \frac{3}{(n+1)!}$$

其中利用了不等式 $\mathrm{e}^0 \leqslant \mathrm{e}^c \leqslant \mathrm{e}^1$. 作为一个实际的数值例子, 假如要用 $p_n(1)$ 逼近 e, 使得误差满足

$$R_n(1) \leqslant 10^{-9}$$

利用 $R_n(1)$ 的上界, 只需要使上界满足

$$\frac{3}{(n+1)!} \leqslant 10^{-9}$$

当 $n \geqslant 12$ 时上式成立, 因此 $p_{12}(1)$ 是 e 的一个满足精度要求的逼近. ■

对大学数学中遇到的大部分常见函数, 公式 (1.6) 和 (1.11) 提供了多项式逼近和相应的余项公式. 为了后面的需要, 我们给出一些较重要的公式.

$$\mathrm{e}^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \frac{x^{n+1}}{(n+1)!}\mathrm{e}^c \tag{1.13}$$

$$\begin{aligned}\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots &+ (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!} \\ &+ (-1)^n \frac{x^{2n+1}}{(2n+1)!}\cos(c)\end{aligned} \tag{1.14}$$

$$\begin{aligned}\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots &+ (-1)^n \frac{x^{2n}}{(2n)!} \\ &+ (-1)^{n+1}\frac{x^{2n+2}}{(2n+2)!}\cos(c)\end{aligned} \tag{1.15}$$

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots + x^n + \frac{x^{n+1}}{1-x}, \quad x \neq 1 \tag{1.16}$$

$$\begin{aligned}(1+x)^\alpha = 1 + \binom{\alpha}{1}x + \binom{\alpha}{2}x^2 + \cdots &+ \binom{\alpha}{n}x^n \\ &+ \binom{\alpha}{n+1}x^{n+1}(1+c)^{\alpha-n-1}\end{aligned} \tag{1.17}$$

在最后一个公式中, α 是任意实数. 系数 $\binom{\alpha}{k}$ 称为**二项式系数**, 它由下式定义

$$\binom{\alpha}{k} = \frac{\alpha(\alpha-1)\cdots(\alpha-k+1)}{k!}, \quad k = 1, 2, 3, \cdots, \quad \binom{\alpha}{0} = 1$$

除了 (1.16) 式外, 所有公式中的点 c 在 0 和 x 之间. 公式 (1.16) 的证明见习题 10.

例 1.2.4 当 $|x| \leqslant \pi/4$ 时, 构造 $\cos(x)$ 的泰勒多项式, 要求逼近误差不超过 10^{-5}. 因为在 (1.15) 的余项中的点 c 是未知的, 所以要考虑最坏的可能情况, 使它满足所要求的误差界:

$$|R_{2n+1}(x)| \leqslant \frac{x^{2n+2}}{(2n+2)!} \leqslant 10^{-5}, \quad |x| \leqslant \pi/4$$

这里使用了关系式 $|\cos(c)| \leqslant 1$. 为使上述不等式成立, 必须有

$$\frac{(\pi/4)^{2n+2}}{(2n+2)!} \leqslant 10^{-5}$$

当 $n \geqslant 3$ 时上式满足. 所以所要求的逼近是

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}$$ ■

许多泰勒多项式和余项并不是直接从 (1.6) 和 (1.11) 得到的. 常常可以巧妙地利用常见函数的泰勒公式得到其他函数的泰勒多项式逼近. 例如, 考虑构造 $f(x) = \mathrm{e}^{-x^2}$ 关于 $x = 0$ 的泰勒多项式逼近. 先在 (1.13) 中用 t 代替 x, 得到

$$\mathrm{e}^t \approx 1 + t + \frac{t^2}{2!} + \cdots + \frac{t^n}{n!} + \frac{t^{n+1}}{(n+1)!}\mathrm{e}^c$$

其中 c 是 0 和 t 之间的未知数. 上式对一切实数 t 成立. 现在用 $-x^2$ 代替 t, 得到

$$\mathrm{e}^{-x^2} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \cdots + \frac{(-1)^n x^{2n}}{n!} + \frac{(-1)^{n+1} x^{2n+2}}{(n+1)!}\mathrm{e}^c \tag{1.18}$$

其中 c 是满足 $-x^2 \leqslant c \leqslant 0$ 的未知数. 这是逼近 e^{-x^2} 的 $2n$ 次 (同时也是 $2n+1$ 次) 的泰勒多项式. 如果按 1.1 节的方式直接构造这个泰勒逼近, 我们会发现 e^{-x^2} 的高阶导数难以计算.

作为间接构造泰勒多项式逼近的一个更加复杂的例子, 我们推导 $\ln(1-t)$ 的逼近. 公式 (1.16) 两边关于 x 从 0 到 t 积分:

$$\int_0^t \frac{\mathrm{d}x}{1-x} = \int_0^t (1 + x + x^2 + \cdots + x^n)\mathrm{d}x + \int_0^t \frac{x^{n+1}}{1-x}\mathrm{d}x$$

$$-\ln(1-t) = t + \frac{1}{2}t^2 + \frac{1}{3}t^3 + \cdots + \frac{1}{n+1}t^{n+1} + \int_0^t \frac{x^{n+1}}{1-x}\mathrm{d}x$$

$$\ln(1-t) = -\left(t + \frac{1}{2}t^2 + \cdots + \frac{1}{n+1}t^{n+1}\right) - \int_0^t \frac{x^{n+1}}{1-x}\mathrm{d}x \tag{1.19}$$

上式对 $0 \leqslant t < 1$ 成立. 应用积分中值定理 (附录 A 中定理 A.8) 可以简化余项, 得到

$$\int_0^t \frac{x^{n+1}}{1-x}\mathrm{d}x = \frac{1}{1-c}\int_0^t x^{n+1}\mathrm{d}x = \left(\frac{1}{1-c}\right)\frac{t^{n+2}}{n+2}$$

上式对 0 和 t 之间某个 c 成立. 上面的论证对 $-1 \leqslant t < 0$ 也成立.

这样, 我们得到

$$\ln(1-t) = -\left(t + \frac{1}{2}t^2 + \cdots + \frac{1}{n+1}t^{n+1}\right) - \left(\frac{1}{1-c_t}\right)\frac{t^{n+2}}{n+2} \tag{1.20}$$

其中 c_t 是 0 和 t 之间的一个未知数, 上式对 $-1 \leqslant t < 1$ 成立.

注记 1.2.5 当提及界定某个量的误差时, 比如说, 界定误差 $f(x) - p_n(x)$, 是指找一个数 M_1 使

$$|f(x) - p_n(x)| \leqslant M_1$$

当提及界定在一个区间 $\alpha \leqslant x \leqslant \beta$ 上的误差时, 是指找一个数 M_2 使

$$\max_{\alpha \leqslant x \leqslant \beta} |f(x) - p_n(x)| \leqslant M_2$$

大部分例子不像例 1.2.3 那样 "简单", 一般情况下必须考虑误差的绝对值. ■

无穷级数

通过重新安排 (1.16) 中的项, 得到一个有限的几何级数之和

$$1 + x + x^2 + \cdots + x^n = \frac{1-x^{n+1}}{1-x}, \quad x \neq 1 \tag{1.21}$$

当 $|x| < 1$ 时, 在 (1.16) 中令 $n \to \infty$, 得到无穷几何级数

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots = \sum_{j=0}^{\infty} x^j, \quad |x| < 1 \tag{1.22}$$

一般地, 如果部分和 $S_n = \sum_{j=0}^{\infty} c_j (n \geqslant 0)$ 构成一个收敛序列, 就称无穷级数

$$\sum_{j=0}^{\infty} c_j$$

收敛. 这意味着

$$S = \lim_{n\to\infty} S_n$$

存在, 且记

$$S = \sum_{j=0}^{\infty} c_j \tag{1.23}$$

对 $x \neq 1$ 时的无穷级数 (1.22), 由 (1.21) 给出其部分和

$$S_n = \frac{1 - x^{n+1}}{1 - x}$$

当 $|x| < 1$ 时, 序列 $\{S_n\}$ 显然有极限值 $1/(1-x)$, 这时就说无穷级数收敛于这个值. 当 $|x| > 1$ 时, 序列 $\{S_n\}$ 显然不收敛于任何极限值, 这时就说无穷几何级数发散. 当 $|x| = 1$ 时级数的收敛情况如何呢?

假设 $f(x)$ 在 $x = a$ 处有任意阶导数. 无穷级数

$$\sum_{j=0}^{\infty} \frac{(x-a)^j}{j!} f^{(j)}(a)$$

称为函数 $f(x)$ 在点 $x = a$ 处的**泰勒级数展开**. 部分和

$$\sum_{j=0}^{n} \frac{(x-a)^j}{j!} f^{(j)}(a)$$

就是泰勒多项式 $p_n(x)$. 如果当 $n \to \infty$ 时误差趋向于零

$$\lim_{n \to \infty} [f(x) - p_n(x)] = 0$$

则序列 $\{p_n(x)\}$ 有极限 $f(x)$. 此时, 我们有等式

$$f(x) = \sum_{j=0}^{\infty} \frac{(x-a)^j}{j!} f^{(j)}(a) \tag{1.24}$$

例如, 对适当的 x 值可以证明, 当 $n \to \infty$ 时 (1.13)~(1.17) 和 (1.20) 中的误差项趋向于零. 这就得到下列泰勒展开式:

$$\mathrm{e}^x = \sum_{j=0}^{\infty} \frac{x^j}{j!}, \quad -\infty < x < \infty \tag{1.25}$$

$$\sin x = \sum_{j=0}^{\infty} \frac{(-1)^j x^{2j+1}}{(2j+1)!}, \quad -\infty < x < \infty \tag{1.26}$$

$$\cos x = \sum_{j=0}^{\infty} \frac{(-1)^j x^{2j}}{(2j)!}, \quad -\infty < x < \infty \tag{1.27}$$

$$(1+x)^\alpha = \sum_{j=0}^{\infty} \binom{\alpha}{j} x^j, \quad -1 < x < 1 \tag{1.28}$$

$$\ln(1-t) = -\sum_{j=0}^{\infty} \frac{t^j}{j}, \quad -1 \leqslant x < 1 \tag{1.29}$$

在 (1.28) 中, 级数可能对其他的 x 值收敛, 不过这样的 x 值与 α 有关.

形式为

$$\sum_{j=0}^{\infty} a_j (x-a)^j \tag{1.30}$$

的无穷级数称为**幂级数**. 泰勒公式是得到这种级数的一种方法, 不过幂级数也可以由别的途径产生. 它们的收敛性可以直接检验而不必依赖于泰勒误差公式. 我们给出用于检验它们收敛性的两个重要定理.

定理 1.2.6 假设对某个 x_0, 级数 (1.30) 收敛. 则级数 (1.30) 也对一切满足 $|x-a| < |x_0 - a|$ 的 x 收敛.

习题 24 给出这个结果证明的梗概.

定理 1.2.7 对级数 (1.30), 假设极限

$$R = \lim_{n \to \infty} \left| \frac{a_{n+1}}{a_n} \right|$$

存在. 则对满足 $|x-a| < 1/R$ 的 x, 级数 (1.30) 收敛于一个极限 $S(x)$. 当 $R=0$ 时, 对任意的实数 x, 级数 (1.30) 都收敛.

例如, 我们来检查公式 (1.27) 中幂级数的收敛性. 设 $t = x^2$, 得到级数

$$\sum_{j=0}^{\infty} \frac{(-1)^j t^j}{(2j)!} \tag{1.31}$$

对

$$a_j = \frac{(-1)^j}{(2j)!}$$

应用定理 1.2.7, 得到 $R=0$. 故级数 (1.31) 对任何 t 的值收敛, 于是, 公式 (1.27) 中的级数对任何 x 的值收敛.

关于这两个定理和无穷级数的详细讨论可参考任何一本初等微积分教材.

习题

1. 在 $[-1,1]$ 上界定用 $p_3(x)$ 逼近 e^x 的误差, $p_3(x)$ 及其余项在 (1.13) 中给出. 把它与表 1.1 中给出的结果作比较.
2. 求 $f(x) = e^x \sin(x)$ 关于点 $a=0$ 的 2 次泰勒多项式. 当 $-\pi/4 \leqslant x \leqslant \pi/4$ 时界定这个逼近的误差.
3. 求 $f(x) = \sqrt[3]{x}$ 关于点 $a=8$ 的线性泰勒多项式逼近和 2 次泰勒多项式逼近. 在区间 $8 \leqslant x \leqslant 8+\delta(\delta>0)$ 上界定每种逼近的误差. 在区间 $[8, 8.1]$ 上找到一个具体的数值界.

4. (a) 对 $-\pi/4 \leqslant x \leqslant \pi/4$, 界定逼近

$$\sin(x) \approx x$$

中的误差.

(b) 对小的 x 值上式是一个好的逼近, 再考察 "百分误差"

$$\frac{\sin(x)-x}{\sin(x)} \approx \frac{\sin(x)-x}{x}$$

对 $-\delta \leqslant x \leqslant \delta$ 界定后面的量的绝对值. 取 δ 使百分误差的绝对值小于 1%.

5. 在 (1.14) 中应该选择次数 $2n-1$ 多大, 才能使得对一切 $-\pi/2 \leqslant x \leqslant \pi/2$ 有

$$|\sin(x) - p_{2n-1}(x)| \leqslant 0.001$$

用 $x=\pi/2$ 处求得的 $p_{2n-1}(x)$ 的值检验你的结果.

6. 设 $p_n(x)$ 是函数 $f(x)=\ln(1-x)$ 关于点 $a=0$ 的 n 次泰勒多项式. 应该选择 n 多大, 使得对 $-1/2 \leqslant x \leqslant 1/2$ 有 $|f(x)-p_n(x)| \leqslant 10^{-4}$? 对 $-1 \leqslant x \leqslant 1/2$ 呢?

7. 对 (1.17) 与 $\alpha = 1/2$, 写出 $p_4(x)$. 界定 $0 \leqslant x \leqslant 1/2$ 时的误差.

8. 在 (1.13) 中应该选择 n 多大, 才有

$$|e^x - p_n(x)| \leqslant 10^{-5}, \quad -1 \leqslant x \leqslant 1$$

9. 用带余项的泰勒多项式求下列极限值:

(a) $\lim\limits_{x\to 0} \dfrac{1-\cos(x)}{x^2}$ (b) $\lim\limits_{x\to 0} \dfrac{\ln(1+x^2)}{2x}$ (c) $\lim\limits_{x\to 0} \dfrac{\ln(1-x)+xe^{x/2}}{x^3}$

提示: 利用常见函数 [例如 $\cos(t), \ln(1+t)$ 和 e^t] 的泰勒多项式得到这些分式的分子的多项式逼近, 然后简化结果.

10. 验证 (1.16).

提示: 用 $1-x$ 乘以两边并简化.

11. 证明

$$(1+t)^n = \sum_{j=0}^{n} \binom{n}{j} t^j$$

12. 把 $f(x) = 1 + x^6$ 改写为形式

$$f(x) = \sum_{j=0}^{6} a_j (x-1)^j$$

提示: 把 $f(x)$ 展开为关于 $x=1$ 的 6 次泰勒多项式. 利用此式, 系数 $\{a_j\}$ 是什么? 此时, 误差 $f(x)-p_6(x)$ 是什么?

13. 在 10^{-6} 的精度内求 $I=\int_0^1 \frac{e^x-1}{x}dx$ 的值.

提示: 用 e^x 的泰勒多项式逼近及余项公式.

14. (a) 找出 $f(t) = 1/(1+t^2)$ 关于 $a=0$ 时带余项的泰勒多项式.

提示: 把 $x=-t^2$ 代入到 (1.16) 中.

(b) 通过下述过程推导 $g(x)=\tan^{-1}x$ 的泰勒多项式及其余项公式. 把 (a) 得到的等式两边积分, 并利用

$$\tan^{-1}(x)=\int_0^x \frac{\mathrm{d}t}{1+t^2}$$

15. 定义 $f(x)=\int_0^x \frac{1-\mathrm{e}^t}{t}\mathrm{d}t$. 求 $f(x)$ 的 2 次泰勒多项式逼近. 给出逼近误差的公式, 并在区间 $0\leqslant x\leqslant\delta$ 上界定误差, 其中 $\delta>0$. 对 $\delta=0.1$ 误差界是什么?

16. 定义 $f(x)=\int_0^x \frac{\ln(1+t)}{t}\mathrm{d}t$.

(a) 给出 $f(x)$ 关于 $x=0$ 的泰勒多项式逼近.

(b) 对 $|x|\leqslant 1/2$ 界定 n 次逼近中的误差.

(c) 求 n 的值, 使得相应的泰勒多项式逼近的误差在 $[-\frac{1}{2},\frac{1}{2}]$ 上至多为 10^{-7}.

17. 定义 $f(x)=\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$.

(a) 利用 (1.13), 给出 $f(x)$ 关于 $x=0$ 的泰勒多项式逼近.

(b) 对 $|x|\leqslant 1$ 界定 n 次逼近中的误差.

(c) 求 n 的值, 使得相应的泰勒多项式逼近的误差在 $[-1,1]$ 上至多为 10^{-7}.

18. 定义 $f(x)=\frac{1}{x}\int_0^x \frac{\mathrm{d}t}{1+t^3}$. 对 $f(x)$ 求泰勒多项式逼近, 逼近次数为 3 或更大些. 给出逼近的误差公式. 估计 $f(0.1)$ 并界定误差.

提示: 利用 $1/(1-u)$ 的泰勒逼近

19. 求

$$f(x)=\ln\left(\frac{1+x}{1-x}\right),\quad -1<x<1$$

关于 $x=0$ 的泰勒多项式.

提示: 记

$$f(t)=\ln(1+t)-\ln(1-t)$$

$$f'(t)=\frac{1}{1+t}+\frac{1}{1-t}=\frac{2}{1-t^2}$$

应用 (1.16) 与 $x=t^2$ 展开 $1/(1-t^2)$. 得到

$$f'(t)=2[1+t^2+t^4+\cdots+t^{2n}]+\frac{2t^{2n+2}}{1-t^2},\quad -1<t<1.$$

通过求

$$f(x)=\int_0^x f'(t)\mathrm{d}t$$

的值来得到 $f(x)$ 的包括误差项的泰勒多项式逼近.

20. (a) 利用习题 19, 给出 $\ln(2)$ 的泰勒多项式逼近并界定误差.

提示: 为了利用习题 19 的结果所需的 x 值是什么?

(b) 对 $\frac{1}{2}\leqslant z\leqslant 1$, 求 $\ln(z)$ 的值. 利用习题 19 给出计算 $\ln(z)$ 的一个方法. 界定误差.

21. (a) 考虑利用

$$\pi=4\tan^{-1}(1)$$

求 π 的值. 利用习题 14 的结果, 为计算精确到 10^{-10} 的 π 的近似值, 在泰勒逼近 $\tan^{-1}(x) \approx p_{2n-1}(x)$ 中需要多少项? 这是求 π 值的一个可行方法吗?

(b) 利用 $\tan^{-1}(x)$ 的级数提出另一个计算 π 的方法, 给出一个更迅速收敛的方法. (关于迅速收敛于 π 的级数有大量的研究文献.)

22. 定义 $h(x) = f(x)g(x)$. 设 $f(x)$ 和 $g(x)$ 的 n 次泰勒多项式分别是

$$p_n(x) = \sum_{i=0}^{n} a_i x^i, \quad q_n(x) = \sum_{j=0}^{n} b_j x^j$$

在乘积 $p_n(x)q_n(x)$ 中去掉所有次数大于 n 的项后得到 $r_n(x)$.

(a) 对 $n = 2$, 证明 $h(x)$ 的 2 次泰勒多项式等于 $r_2(x)$.

(b) 对一般的 $n \geqslant 1$, 证明 $h(x)$ 的 n 次泰勒多项式等于 $r_n(x)$.

提示: 利用莱布尼兹(Leibniz)公式对积 $f(x)g(x)$ 反复求导:

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k}[f(x)g(x)] = \sum_{j=0}^{k} \binom{k}{j} f^{(j)}(x) g^{(k-j)}(x)$$

23. 回顾 (1.23) 的无穷级数

$$S = \sum_{j=0}^{\infty} c_j$$

收敛的定义. 证明级数收敛的一个必要条件是

$$\lim_{j \to \infty} c_j = 0$$

24. 考虑定理 1.2.6 的证明. 因为假定级数 $\sum_{j=0}^{\infty} a_j(x_0 - a)^j$ 对 $x = x_0$ 收敛, 所以由习题 23 的结果可推出

$$\lim_{j \to \infty} a_j(x_0 - a)^j = 0$$

因此, 存在常数 $c > 0$ 使得

$$\left| a_j(x_0 - a)^j \right| \leqslant c, \quad j \geqslant 0$$

引入部分和

$$S_n = \sum_{j=0}^{n} a_j(x - a)^j, \quad n \geqslant 0$$

并定义

$$r = \left| \frac{x - a}{x_0 - a} \right|$$

根据定理的假设, $r < 1$.

(a) 证明部分和 $\{S_n\}$ 满足

$$|S_m - S_n| \leqslant c\frac{r^{n+1}}{1 - r}, \quad 一切 \quad m > n \geqslant 0$$

由此推断

$$\lim_{n,m\to\infty} |S_m - S_n| = 0$$

所以序列 $\{S_n\}$ 构成柯西(Cauchy)序列. 从高等分析的一个定理知, 极限

$$S = \lim_{n\to\infty} S_n$$

存在, 因而证明了 (1.30) 的无穷级数

$$S = \sum_{j=0}^{\infty} a_j (x-a)^j, \quad |x-a| < |x_0 - a|$$

是收敛的.

(b) 证明部分和中的误差满足

$$|S - S_n| \leqslant \frac{cr^{n+1}}{1-r}$$

此式通常用于得到以部分和 S_n 逼近无穷级数 S,

$$S \approx S_n$$

时的误差界.

1.3 多项式求值

多项式求值看起来是一个简单的任务, 但是实际情况并非如此. 为了说明这种可能性, 考虑计算

$$p(x) = 3 - 4x - 5x^2 - 6x^3 + 7x^4 - 8x^5$$

从程序设计员的观点来看, 求值最简单的方法是独立地计算每一项. 确切地说, 对一般项 cx^k, 在程序中用

$$c * x^\wedge k \quad \text{或} \quad c * x * * k$$

虽然有些 "智能的" 编译程序可能产生比较有效的代码, 但是上述方法需要用 k 次乘法. 这个方法求 $p(x)$ 的值将需要

$$1 + 2 + 3 + 4 + 5 = 15 \text{ 次乘法}$$

第二种求值的方法更加有效. 利用前面 x 的幂乘以 x 来计算每个 x 的幂, 如

$$x^3 = x(x^2), \quad x^4 = x(x^3), \quad x^5 = x(x^4) \tag{1.32}$$

因此, 对 $k > 1$, 每个 cx^k 项共做 2 次乘法. $p(x)$ 求值使用

$$1 + 2 + 2 + 2 + 2 = 9 \text{ 次乘法}$$

对高次多项式而言，第二个方法比第一个方法节省许多.

第三种方法称为**嵌套乘法**. 在这个方法中，我们把 $p(x)$ 写成

$$p(x) = 3 + x(-4 + x(-5 + x(-6 + x(7 - 8x))))$$

这样所需的乘法数仅为 5 次，它比第二个方法节省更多. 嵌套乘法是优先考虑的求值过程，当多项式的次数变大时它的优点更加突出.

考虑一般的 n 次多项式

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n, \quad a_n \neq 0 \tag{1.33}$$

若我们使用第二种方法，如 (1.32) 中那样计算 x 的幂，则 $p(x)$ 求值中乘法的次数是 $2n-1$. 对于嵌套乘法，按形式

$$p(x) = a_0 + x(a_1 + x(x_2 + \cdots + x(a_{n-1} + a_n x)\cdots) \tag{1.34}$$

写出 $p(x)$ 并求 $p(x)$ 的值. 这里仅需要 n 次乘法，它比第二种方法大约节省 50%. 所有的方法都需要 n 次加法. 嵌套乘法的用法在 1.1 节末尾给出的 MATLAB 程序中实施.

例 1.3.1 求 $\ln(x)$ 关于 $a=1$ 的泰勒多项式 $p_5(x)$. 在公式 (1.19) 中，用 $-(x-1)$ 代替 t. 由此得到

$$p_5(x) = (x-1) - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 - \frac{1}{4}(x-1)^4 + \frac{1}{5}(x-1)^5$$

设 $w = x-1$ 并记

$$p_5(x) = w\left(1 + w\left(-\frac{1}{2} + w\left(\frac{1}{3} + w\left(-\frac{1}{4} + \frac{1}{5}w\right)\right)\right)\right)$$

在计算机程序中，系数以十进制形式存放. ■

我们以形式 (1.34) 为基础给出一个正式的算法. 假设要求 $p(x)$ 在某个数 z 处的值. 定义系数 b_i 的一个序列如下：

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + zb_n \\ b_{n-2} &= a_{n-2} + zb_{n-1} \\ &\vdots \\ b_0 &= a_0 + zb_1 \end{aligned} \tag{1.35}$$

则

$$p(z) = b_0 \tag{3.36}$$

系数 b_j 代表 (1.34) 中某一对括号内的值. b_{n-1} 是最内层括号里表达式的值, 而 b_0 是整个多项式的值. 按这种形式计算, 嵌套乘法称为**霍纳**(Horner)**法**; 它与许多初等代数教材中讨论的**综合除法**(synthetic division) 密切相关.

利用 (1.35) 得到的系数, 定义多项式

$$q(x) = b_1 + b_2x + b_3x^2 + \cdots + b_nx^{n-1} \tag{1.37}$$

可以证明

$$p(x) = b_0 + (x - z)q(x) \tag{1.38}$$

因此, $q(x)$ 是用 $x - z$ 除 $p(x)$ 的商, b_0 是余数. (1.38) 的证明作为习题 7 留给读者. 研究多项式求根方法时, 若一个根 z 已经求出, 使用上述结果就可以降低多项式的次数, 因为 $b_0 = 0$, 所以 $p(x) = (x - z)q(x)$.

一个例题程序

本节的最后, 我们给出函数

$$\text{Sint}\ x = \frac{1}{x}\int_0^x \frac{\sin(t)}{t}\text{d}t, \quad x \neq 0 \tag{1.39}$$

的泰勒多项式近似求值的 MATLAB 程序, 其中 $\text{Sint}(0) = 1$. 这个函数称为**正弦积分**. 对 $-6 \leqslant x \leqslant 6$, $\text{Sint}\ x$ 的图形在图 1-5 中给出.

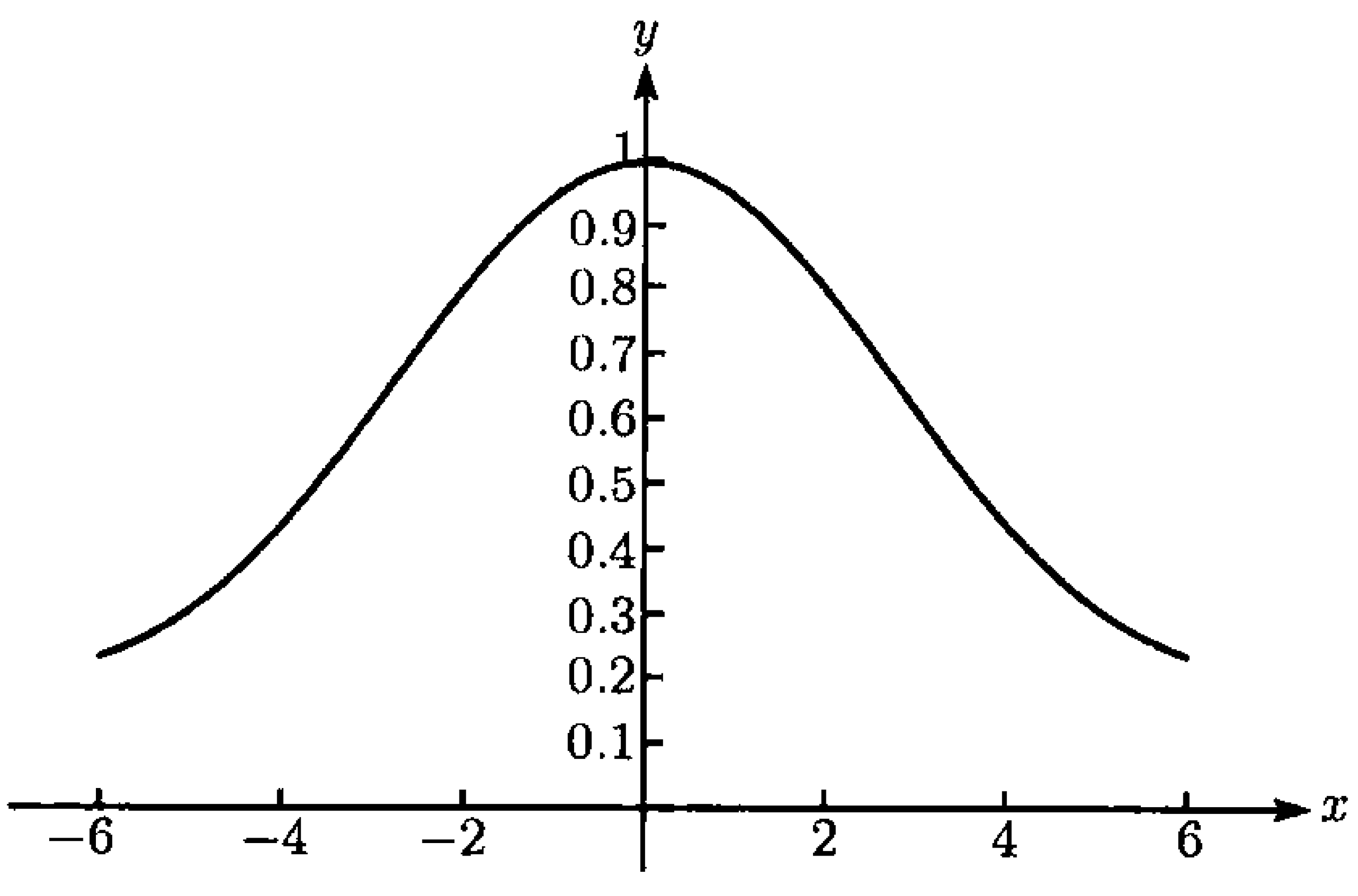


图 1-5　(1.39) 的正弦积分 Sint (x)

先构造 $\text{Sint}\ x$ 在 $[-1, 1]$ 上的泰勒多项式, 要求在 $[-1, 1]$上它的最大误差不超过 5×10^{-9}, 区间 $[-1, 1]$ 和误差界 5×10^{-9} 是一个相当任意的选择. 我们利用 (1.14) 给出的 $\sin(t)$ 的级数公式, 用 t 替代 x. 首先考虑 $x > 0$ 的情形. 两边除以 t 并在 $[0, x]$ 上积分得到

$$\text{Sint}\ x = \frac{1}{x}\int_0^x \left[1 - \frac{t^2}{3!} + \frac{t^4}{5!} - \cdots + (-1)^{n-1}\frac{t^{2n-2}}{(2n-1)!}\right]\text{d}t + R_{2n-2}(x)$$

$$= 1 - \frac{x^2}{3!3} + \frac{x^4}{5!5} - \cdots + (-1)^{n-1}\frac{x^{2n-2}}{(2n-1)!(2n-1)} + R_{2n-2}(x) \quad (1.40)$$

$$R_{2n-2}(x) = \frac{1}{x}\int_0^x (-1)^n \frac{t^{2n}}{(2n+1)!}\cos(c_t)\mathrm{d}t$$

其中点 c_t 在 0 和 t 之间. 因为 $|\cos(c_t)| \leqslant 1$, 所以

$$|R_{2n-2}(x)| \leqslant \frac{1}{x}\int_0^x \frac{t^{2n}}{(2n+1)}\mathrm{d}t = \frac{x^{2n}}{(2n+1)!(2n+1)} \quad (1.41)$$

容易看出, 对 $x < 0$ 这个界也成立. 按照要求, 选择次数使得

$$|R_{2n-2}(x)| \leqslant 5 \times 10^{-9} \quad (1.42)$$

由 (1.41),

$$\max_{|x|\leqslant 1} |R_{2n-2}(x)| \leqslant \frac{1}{(2n+1)!(2n+1)}$$

选择 n 使得这个上界不超过 5×10^{-9}. 当 $2n+1 \geqslant 11$, 即 $n \geqslant 5$ 时这个结论也成立, 因此满足 (1.42). 我们使用的多项式是

$$p(x) = 1 - \frac{x^2}{3!3} + \frac{x^4}{5!5} - \frac{x^6}{7!7} + \frac{x^8}{9!9}, \quad -1 \leqslant x \leqslant 1 \quad (1.43)$$

作为 $[-1, 1]$ 上 Sint x 的一个逼近, 它的误差满足 (1.42).

上面的多项式 $p(x)$ 是一个偶函数, 即对一切 x, $p(-x) = p(x)$, 偶次多项式只含 x 的偶次幂 (参见习题 3). 在 $p(x)$ 中利用 $u = x^2$ 得到 4 次多项式

$$g(u) = 1 - \frac{u}{18} + \frac{u^2}{600} - \frac{u^3}{35\,280} + \frac{u^4}{3\,265\,920} \quad (1.44)$$

通过求 $g(u)$ 的值可以更有效地求 $p(x)$ 的值.

MATLAB 程序: Sint x的泰勒多项式 Sint x 关于 0 的泰勒多项式 $p_n(x)$ 是偶次多项式. 类似于从 (1.43) 到 (1.44) 的处理方法, 在求值中可以减少大约一半的乘法次数. 这里给出 Sint x 的泰勒逼近求值的 MATLAB 程序`plot_Sint`. 泰勒多项式具有四个不同的次数 n(这里给出的是 2, 4, 6, 8), 并且在用户指定的区间 [0, b] 上求值.

```
% TITLE: Plot Taylor polynomials for the ''sine integral''
% about x = 0.
%
% This plots several Taylor polynomials and their errors
% for increasing degrees.  The particular function being
% approximated is Sint(x) on [0,b], with x = 0 the point of
% expansion for creating the Taylor polynomials.  We plot
% the Taylor polynomials for several degrees, which can
% be altered by the user.  Note that the function being
% plotted is symmetric about x=0.
%
% The Taylor polynomials in this case contain only terms
% of even degree.  Such polynomials are called '' even
% polynomials,'' and they are symmetric about x=0.
%
% TO THE STUDENT: run this program for various input values
% of '' b.''  Also, change the values given in the vector
% '' degree'' given below, to experiment with different
% degrees of Taylor polynomial approximations.

% Initialize
b = input('Give the number b defining the interval [0,b] ');
h = b/200;
x = 0:h:b;
max_degree = 20;

% Produce the Taylor coefficients for the ''sine integral''
% function ''Sint.''
c = sint_tay(max_degree);

% Specify the four values of degree to be considered.  They must
% all be even, and they must be less than or equal to max_degree.
degree = [2, 4, 6, 8];
if max(degree) > max_degree
  fprintf('Some value of degree is greater than max_degree =...
         %2.0f\n',max_degree)
  return

end

% Initialize the array to contain the polynomial values.  Row #i
% is to contain the values for the polynomial of degree=degree(i).
p = zeros(4,length(x));

% Calculate the Taylor polynomials
for i = 1:4
  p(i,:)  = poly_even(x,c,degree(i));
end
```

```
% Initialize for plotting the Taylor polynomials
hold off
clf
axis([0,b,0,1])
hold on

% Plot the Taylor polynomials
plot(x,p(1,:),x,p(2,:),':',x,p(3,:),'--',x,p(4,:),'-.')
plot([0,b],[0,0])
plot([0 0],[0 1])
title('Taylor approximations of Sint(x)')
text(1.025*b,0,'x')
text(0,1.03,'y')
legend(strcat('degree = ',int2str(degree(1))),...
       strcat('degree = ',int2str(degree(2))),...
       strcat('degree = ',int2str(degree(3))),...
       strcat('degree = ',int2str(degree(4))))
```

使用下列取名为`sint_tay`的程序对 Sint x 计算 (1.40) 的泰勒系数.

```
function coeff=sint_tay(n)
%
% function coeff = sint_tay(n)
%
% Evaluate the coefficients of the Taylor approximation
% of degree n which approximates the '' sine integral.''
% The input variable n must be an even integer.  The
% output vector coeff will have length m+1 where m=n/2.
% This is because only the even ordered coefficients are
% nonzero, and this must be considered in evaluating the
% associated Taylor polynomial.

m = double(int32(n/2));
if n ~= 2*m
  disp('Error in poly_even(x,coeff,n):')
  disp('The parameter n must be an even integer.')
end

coeff = ones(m+1,1);
sign = 1;
fact = 1;

for i = 2:m+1
  sign = -sign;
  d = 2*i-1;
  fact = fact*(d-1)*d;
```

```
  coeff(i) = sign/(fact*d);
end
```

程序plot_sint也用到下列取名为plot_even的程序求偶次多项式逼近 Sint x 的值. 这是本节前面讨论的程序polyeval的变形, 其中利用了 (1.44) 中所带来的简化.

```
function value = poly_even(x,coeff,n);
%
% function value = poly_even(x,coeff,n)
%
% Evaluate an ''even'' Taylor polynomial at the points given
% in x, with n the degree of the polynomial.  The coefficients
% are to be given in coeff.  It is assumed that the numbers in
% coeff are the nonzero coefficients of the even-ordered
% terms in the polynomial.  The input parameter n must be an
% even integer.

m = double(int32(n/2));
if n ~= 2*m
  disp('Error in poly_even(x,coeff,n):')
  disp('The parameter n must be an even integer.')
end

xsq = x.*x;
value = coeff(m+1)*ones(size(x));

for i = m:-1:1
  value = coeff(i) + xsq.*value;
end
```

习题

1. (a) 实施函数plot_sint, 并把它与在区间 [0, 1], [0, 2], [0, 3] 以及 [0, 4] 上不同次数的逼近作比较. 尽最大努力评述逼近的精确度.

   (b) 修改程序以得到更高次数的逼近, 例如, 在程序中使用
   $$\text{degree} = [4, 8, 12, 16]$$
   重复 (a) 并且也考虑更长的区间 [0, b].
2. 重复对 Sint x 产生一个有误差上界 10^{-9} 的多项式逼近过程, 但使用区间为 $-2 \leqslant x \leqslant 2$.
3. (a) 设多项式 $p(x)$ 是一个偶函数, 即对一切相关的 x, $p(-x) = p(x)$. 证明: 这可推出一切奇次项系数为零.

   (b) 设多项式 $p(x)$ 是一个奇函数, 即对一切相关的 x, $p(-x) = -p(x)$. 证明: 这可推出一切偶次项的系数为零.

   (c) 设 $p(x) = a_0 + a_1x + a_2x^2 + a_5x^5$. 给出系数 $\{a_0, a_1, a_2, a_5\}$ 使 $p(x)$ 为偶函数的条件. 重复使 $p(x)$ 为奇函数的情况.

4. 类似于对 Sint x 那样, 准备一个

$$\text{Cint } x = \frac{1}{x}\int_0^x \frac{1-\cos(t)}{t^2}\mathrm{d}t, \quad -1 \leqslant x \leqslant 1$$

求值的 MATLAB 程序. 利用与 Sint x 同样的误差容限, 如同 (1.42) 中那样. 作为比较, $\text{Cint}(1) \doteq 0.486\ 385\ 376\ 235$. 在 $[-1,1]$ 上作 Cint x 的图形.

5. 在概率论中误差函数

$$\text{erf } x = \frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$$

是有用的. 求它的泰勒多项式, 使得对给定的 $b>0$, $|x| \leqslant b$, 误差以 10^{-9} 为界. 指出如何有效地计算泰勒多项式. 在 $[-b,b]$ 上画多项式的图形. 使用 b=1, 3, 5.

6. 假如用 $x-1$ 除 $p(x)=4x^7-3x^6-2x^5+x^4+x^3-1$. 余项是什么? 商是什么?

7. 证明 (1.38). 代入 (1.37), 并把相同的 x 幂集中在一起来计算 $b_0+(x-z)q(x)$. 然后利用 (1.35) 简化那些系数. 先试着对低次的情形 (比如说 $n=3$) 证明, 可能会比较容易.

8. 证明: 在 (1.38) 中 $p'(z)=q(z)$.

9. 尽可能有效地计算

$$p(x) = 1 - \frac{x^3}{3!} + \frac{x^6}{6!} - \frac{x^9}{9!} + \frac{x^{12}}{12!} - \frac{x^{15}}{15!}$$

的值. 需要多少次乘法? 假定所有的系数都已算出并存放.

10. 指出如何有效地计算函数

$$f(x) = 2\mathrm{e}^{4x} - \mathrm{e}^{3x} + 5\mathrm{e}^{x} + 1$$

的值.

提示: 考虑取 $z=\mathrm{e}^x$.

11. 对 $f(x)=\mathrm{e}^x$, 在 $[-1,1]$ 上求误差至多为 10^{-7} 的泰勒逼近. 利用这个逼近, 编写求 e^x 值的函数程序. 把它与 MATLAB 函数`exp(x)`得到的 e^x 的值作比较, 在 $[-1,1]$ 中的 21 个均匀分布的点上比较你的逼近和`exp(x)`之间的差别.

第 2 章　误差和计算机算术运算

大部分数值分析涉及怎样在数值上求解一个问题, 即如何产生一系列能给出满意答案的数值计算步骤. 这个过程当中要考虑这些计算产生的误差, 有算术运算中产生的误差, 也有其他来源的误差. 整本书中考虑各种问题的数值解时, 我们都会同时考虑各种计算过程中的误差. 本章中给出一些关于误差的一般性结论.

2.1 节中考虑当代计算机中数的表示及其产生的后果. 2.2 节包含关于误差的基本定义及其某些重要例子. 2.3 节讨论算术运算中误差的传播. 2.4 节考察某些求和过程中包含的误差并说明数值计算中基本的机器算术运算定义的重要性.

2.1　浮　点　数

必须把数存放在计算机中并对这些数进行算术运算. 大部分计算机有两种方式存放数: 整数格式和浮点格式. 整数格式是相对易懂的, 这里不讨论它. 浮点格式是更一般的格式, 允许存放不是整数的数. 我们在本节中定义浮点格式. 大部分计算机使用二进制数, 在附录 E 中有初步介绍. 我们将讨论当今二进制计算机使用的最流行的浮点格式, 但我们先从十进制数的浮点格式入手.

为了简化说明数的浮点表示, 首先考虑按十进制写出的非零数 x. 它能以唯一的方式被写成

$$x = \sigma \cdot \bar{x} \cdot 10^e \tag{2.1}$$

其中 $\sigma = +1$ 或 -1, e 是一个整数, 而 $1 \leqslant \bar{x} < 10$. 这 3 个量分别称为表达式 (2.1) 的*符号*、*指数*和*有效数字*. 例如

$$124.62 = (1.246\,2) \cdot 10^2$$

具有符号 $\sigma = +1$, 指数 $e = 2$ 和有效数字 $\bar{x}$ =1.246 2. (2.1) 的格式在高中数学教材中通常称为*科学计数法*(scientific notation). 注意, 在许多教材中, *有效数字*(significand) 也称为*尾数*(mantissa).

数 x 的十进制浮点表示基本上是 (2.1) 中给出的形式, 只是对 $\bar{x}$ 中的数字个数和 e 的大小有限制. 例如, 假设限定 $\bar{x}$ 中的数字个数为 4, 而 e 在 -99 和 99 之间. 具有这样一种表示的计算机有 4 位十进制浮点运算. 由于对 $\bar{x}$ 长度的限制, 我们不能准确地存放 $\bar{x}$ 中多于 4 位的数字, 甚至 $\bar{x}$ 的第 4 位通过舍入 (后面我们更精确

地定义) 都有可能需要改变. 一些流行的计算器使用十位十进制浮点运算. 因为对大多数人而言, 十进制算术运算更为直观, 所以我们偶尔利用十进制浮点运算而不是二进制浮点运算来说明一些想法.

现在考虑按二进制格式写出的数 x. 类似于 (2.1), 可记

$$x = \sigma \cdot \bar{x} \cdot 2^e \tag{2.2}$$

其中 $\sigma = +1$ 或 -1, e 是一个整数, $\bar{x}$ 是二进制小数, 它满足

$$(1)_2 \leqslant \bar{x} < (10)_2 \tag{2.3}$$

在十进制中, $1 \leqslant \bar{x} < 2$. 例如, 若 $x = (11\ 011.011\ 1)_2$, 则 $\sigma = +1$, $e = 4 = (100)_2$, $\bar{x} = (1.101\ 101\ 11)_2$. 注意, 对 $x \neq 0$, $\bar{x}$ 中二进制小数点左边的数字保证为 1.

形如 (2.2) 的二进制数 x 的浮点表示对 $\bar{x}$ 中的二进制数字个数和对 e 的大小有限制. $\bar{x}$ 中允许的二进制数字个数称为 x 的二进制浮点表示的**精度**(precision). IEEE 浮点算术运算标准是几乎所有当前计算机中使用的浮点数格式. 例如, Intel 处理机全部使用这个标准. 对于这个标准, x 的 IEEE**单精度浮点表示**有 24 个二进制数字的精度, 而指数 e 的范围为 $-126 \leqslant e \leqslant 127$.

$$x = \sigma \cdot (1.a_1 a_2 \cdots a_{23}) \cdot 2^e \tag{2.4}$$

在二进制中,

$$-(1\ 111\ 110)_2 \leqslant e \leqslant (1\ 111\ 111)_2$$

x 的 IEEE**双精度浮点表示**有 53 个二进制数字的精度, 而指数 e 的范围为 $-1\ 022 \leqslant e \leqslant 1\ 023$.

$$x = \sigma \cdot (1.a_1 a_2 \cdots a_{52}) \cdot 2^e \tag{2.5}$$

单精度浮点格式使用四个字节 (32 位), 其存储方案在表 2-1 中简述. 符号 σ 存放在 b_1 中 (对 $\sigma = +1, b_1 = 0$, 而对 $\sigma = -1, b_1 = 1$). 定义 $E = e + 127$. 位 b_2 到位 b_9 中存放正二进制整数 E 而不是 e. $\bar{x}$ 中的数字 $a_1 a_2 \cdots a_{23}$ 存放在位 b_{10} 到位 b_{32} 中. 当数 x 存放在存储器中时, 浮点表示中 $\bar{x}$ 的首个二进制数字 1 是不存放的; 而当一个浮点数 x 从存储器中取出并放入算术寄存器中进一步使用 (或等价地做某些事情) 时首位数字 1 被嵌入到 $\bar{x}$ 中. 这导致数 $x = 0$ 需要一个特殊的表示, 它按照 $E = 0$, 其中 $\sigma = 0$, $b_{10} b_{11} \cdots b_{32} = (00 \cdots 0)_2$ 来存放. 所有可能的单精度浮点数在表 2-2 中给出. 这个表和表 2-4 选自 Overton 的优秀教材 [27, 表 4-1 和表 4-2].

表 2-1 IEEE 单精度浮点格式的存储

$\underbrace{b_1}_{\sigma}$	$\underbrace{b_2 b_3 \cdots b_9}_{E}$	$\underbrace{b_{10} b_{11} \cdots b_{32}}_{\bar{x}}$

表 2-2 IEEE 单精度浮点格式

$\pm$	$c_1 \cdots c_8$	$a_1 a_2 \cdots a_{23}$

$E=(c_1\cdots c_8)_2$	x
$(00\ 000\ 000)_2=(0)_{10}$	$\pm(0.a_1a_2\cdots a_{23})_2\cdot 2^{-126}$
$(00\ 000\ 001)_2=(1)_{10}$	$\pm(1.a_1a_2\cdots a_{23})_2\cdot 2^{-126}$
$(00\ 000\ 010)_2=(2)_{10}$	$\pm(1.a_1a_2\cdots a_{23})_2\cdot 2^{-125}$
$\vdots$	$\vdots$
$(01\ 111\ 111)_2=(127)_{10}$	$\pm(1.a_1a_2\cdots a_{23})_2\cdot 2^{0}$
$(10\ 000\ 000)_2=(128)_{10}$	$\pm(1.a_1a_2\cdots a_{23})_2\cdot 2^{1}$
$\vdots$	$\vdots$
$(11\ 111\ 101)_2=(253)_{10}$	$\pm(1.a_1a_2\cdots a_{23})_2\cdot 2^{126}$
$(11\ 111\ 110)_2=(254)_{10}$	$\pm(1.a_1a_2\cdots a_{23})_2\cdot 2^{127}$
$(11\ 111\ 111)_2=(255)_{10}$	若$a_1=\cdots=a_{23}=0, \pm\infty$; 否则, NaN

双精度浮点格式使用 8 个字节 (64 位), 在表 2-3 中简述如何存放它. 符号 σ 存放在 b_1 中 (对 $\sigma=+1, b_1=0$, 对 $\sigma=-1, b_1=1$). 定义 $E=e+1\ 023$. 位 b_2 到位 b_{12} 中存放正二进制整数 E 而不是 e. $\bar{x}$ 中的数字 $a_1a_2\cdots a_{52}$ 存放在位 b_{13} 到位 b_{64} 中. 当数 x 存放在存储器中时, 浮点表示中 $\bar{x}$ 的首个二进制数字 1 是不存放的; 而当一个浮点数 x 从存储器中取出并放入算术寄存器中进一步使用时, 首位数字 1 被嵌入到 $\bar{x}$ 中. 这导致数 $x=0$ 需要一个特殊的表示, 它按照 $E=0$, 其中 $\sigma=0$, $b_{13}b_{14}\cdots b_{64}=(00\cdots 0)_2$ 来存放. 所有可能的双精度浮点数在表 2-4 中给出.

此外, 还有正无穷大 ∞ 和负无穷大 $-\infty$ 的计算机表示. 因而, 这样的浮点算术运算中, $\frac{1}{0}=\infty$. 还有一个表示 NaN, 意指 "不定值". 例如 $\frac{0}{0}$ 那样的浮点运算导致答案 NaN, 而不是用某个含义模糊的差错信息使程序突然停止. IEEE 标准有一个附加的细微差别. 当首位数字 1 在 (2.4) 和 (2.5) 中消失时, 相当于表 2-2 和表 2-4 中 $E=0$, 我们称这个浮点格式为**非正规化的**(unnormalized). 与 (2.4) 和 (2.5) 中**正规化的**(normalized)**格式**相比, 它的有效数字精度较低, 因此本教材的大部分内容不涉及它.

表 2-3 IEEE 双精度浮点格式的存储

$\underbrace{b_1}_{\sigma}$	$\underbrace{b_2b_3\cdots b_{12}}_{E}$	$\underbrace{b_{13}b_{14}\cdots b_{64}}_{\bar{x}}$

例 2.1.1 可用 MATLAB 生成表 2-4 的元素. 执行指令

```
format hex
```

将使所有随后的数字以十六进制格式 (基 16) 输出到屏幕上. 例如, 输入数字 7 将产生输出

$$4\ 01c\ 000\ 000\ 000\ 000 \tag{2.6}$$

16 个十六进制数字是{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}. 为了得到二进制表示, 可利用表 2-5 把每个十六进制数字转换成 4 位二进制数字. 对上面的数字, 得到二进制展开

$$0\ 100\ 000\ 000\ 011\ 100\ 000\ 0\cdots 0\ 000 \tag{2.7}$$

这是 IEEE 双精度浮点格式中的数 7. ■

表 2-4 IEEE 双精度格式

$\pm$	$c_1\cdots c_{11}$	$a_1a_2\cdots a_{52}$

$E=(c_1\cdots c_{11})_2$	x
$(00\ 000\ 000\ 000)_2=(0)_{10}$	$\pm(0.a_1a_2\cdots a_{52})_2\cdot 2^{-1\ 022}$
$(00\ 000\ 000\ 001)_2=(1)_{10}$	$\pm(1.a_1a_2\cdots a_{52})_2\cdot 2^{-1\ 022}$
$(00\ 000\ 000\ 010)_2=(2)_{10}$	$\pm(1.a_1a_2\cdots a_{52})_2\cdot 2^{-1\ 021}$
$\vdots$	$\vdots$
$(01\ 111\ 111\ 111)_2=(1\ 023)_{10}$	$\pm(1.a_1a_2\cdots a_{52})_2\cdot 2^{0}$
$(10\ 000\ 000\ 000)_2=(1\ 024)_{10}$	$\pm(1.a_1a_2\cdots a_{52})_2\cdot 2^{1}$
$\vdots$	$\vdots$
$(11\ 111\ 111\ 101)_2=(2\ 045)_{10}$	$\pm(1.a_1a_2\cdots a_{52})_2\cdot 2^{1\ 022}$
$(11\ 111\ 111\ 110)_2=(2\ 046)_{10}$	$\pm(1.a_1a_2\cdots a_{52})_2\cdot 2^{1\ 023}$
$(11\ 111\ 111\ 111)_2=(2\ 047)_{10}$	若 $a_1=\cdots=a_{52}=0, \pm\infty;$, 否则$NaN$

表 2-5 十六进制到二进制的转换

16 进制数字	等价的 2 进制数字
0	0000
1	0001
2	0010
$\vdots$	$\vdots$
9	1001
a	1010
$\vdots$	$\vdots$
f	1111

2.1.1 浮点表示的精度

考虑如何以浮点表示精确地存放一个数字. 有很多方法度量浮点数精度, 其中机器ε(machine epsilon) 是最流行的. 对任何特定的浮点格式, 机器 ε 是 1 与按那个格式可存放的大于 1 的最小数之间的差. 单精度 IEEE 格式中, 大于 1 的最小的二进制数是

$$1.000\ 000\ 000\ 000\ 000\ 000\ 000\ 01 \tag{2.8}$$

最后的二进制数字 1 在二进制小数点右边第 23 个位置上. 因而, **单精度IEEE格式中机器ε** 是 2^{-23}. 例如, 由此可得 IEEE 单精度格式不能精确地存放数 $1+2^{-24}$. 由

$$2^{-23} \doteq 1.19 \times 10^{-7} \tag{2.9}$$

可知, 就十进制格式而言, 可用 IEEE 单精度格式近似地存放 7 位十进制数字. 类似地, **双精度IEEE格式中的机器ε** 是 $2^{-52} = 2.22 \times 10^{-16}$, 可用 IEEE 双精度格式近似地存放 16 位十进制数字. MATLAB 中有机器 ε, 它是取名为eps的常数.

度量浮点数精度的另一种方法是寻找最大的整数 M, 使得任何满足 $0 \leqslant x \leqslant M$ 的整数 x 都可以按浮点格式存放或精确地表示. 因为 x 是整数, 所以必将把它转换成 (2.2) 中具有正指数 e 的二进制数 $\bar{x}$. 若 n 是有效数字中二进制数字个数, 则易见可以精确地存放小于或等于

$$(1.11\cdots1)_2 \cdot 2^{n-1}$$

的所有整数, 其中这个有效数字包含 n 位全部为 1 的二进制数字. 这是 n 个连续的 1 组成的整数; 利用 (1.21), 可以算出这个数等于 2^n-1. 此外, $2^n = (1.0\cdots0)_2 \cdot 2^n$ 也可精确地存放. 但是有效数字中没有足够的数位去存放 2^n+1, 因为这将需要 $n+1$ 位二进制数字. 用 n 位二进制浮点表示,

$$M = 2^n \tag{2.10}$$

按这个浮点表达式可以精确地表示任何 $\leqslant M$ 的正整数.

IEEE 单精度格式中任何不超过

$$M = 2^{24} = 16\ 777\ 216$$

的 7 位十进制整数都能精确地存放. IEEE 双精度格式中任何不超过

$$M = 2^{53} \doteq 9.0 \times 10^{15}$$

的 15 位十进制整数和绝大部分 16 位十进制整数都能精确地存放.

2.1.2 舍入和截断

设浮点表示中的有效数字为 n 位二进制数字. 若按 (2.2) 表示的数 x 的有效数字 $\bar{x}$ 有多于 n 位的二进制数字, 则把 x 存放到计算机中时, 必须把 $\bar{x}$ 缩短. 一般地, 这与计算机或浮点算术运算的 IEEE 标准的选择有关, 通常按两种方法之一来执行. 最简单的方法是截断 $\bar{x}$ 到 n 位二进制数字, 略去其余的数字. 第二种方法是根据 $\bar{x}$ 的 n 位后面部分的大小舍入 $\bar{x}$ 到 n 位. 更确切地说, 若 $n+1$ 位上数字为零, 则截断 $\bar{x}$ 到 n 位; 否则截断 $\bar{x}$ 到 n 位并在结果的最后一个数字加上 1. 不管使

用截断还是舍入, 我们将用 $\mathrm{fl}(x)$ 表示数 x 的机器浮点形式(machine floating-point version).

可以证明, 能把数 $\mathrm{fl}(x)$ 写成形式

$$\mathrm{fl}(x) = x \cdot (1+\varepsilon) \tag{2.11}$$

其中 ε 是与 x 有关的一个非常小的数. 因为 ε 很小, 所以 $\mathrm{fl}(x)$ 可看作是 x 的一个微小的扰动. 若使用截断, 则

$$-2^{n+1} \leqslant \varepsilon \leqslant 0 \tag{2.12}$$

若使用舍入, 则

$$-2^{-n} \leqslant \varepsilon \leqslant 2^{-n} \tag{2.13}$$

截断最重要的特征是: (1) 最坏可能的误差是使用舍入时误差的 2 倍; (2) 误差 $x-\mathrm{fl}(x)$ 的符号与 $\mathrm{sign}\, x$ 相同. 后面的特征是两个特征中最糟糕的. 在许多计算中, 它完全排除误差对消的可能性, 例子将在 2.4 节中给出. 对于舍入, 最坏可能的误差只是截断误差的 50%. 更重要的是, 误差 $x-\mathrm{fl}(x)$ 有 50% 的情况是负的, 50% 的情况是正的. 在包含许多算术运算的计算中, 舍入有较好的误差传播行为.

关于单精度 IEEE 浮点算术运算, 可以得到几种舍入和截断的变形. 上面描述的截断方法称为 “舍入归零”, 对此我们有

$$-2^{-23} \leqslant \varepsilon \leqslant 0 \tag{2.14}$$

大多数 IEEE 标准用户使用 (2.11) 描述的舍入. 有效数字中有 $n=24$ 位数字, 用标准的舍入,

$$-2^{-24} \leqslant \varepsilon \leqslant 2^{-24} \tag{2.15}$$

双精度 IEEE 浮点算术运算相应的结果是

$$\begin{aligned} &-2^{-52} \leqslant \varepsilon \leqslant 0, && \text{截断} \\ &-2^{-53} \leqslant \varepsilon \leqslant 2^{-53}, && \text{舍入} \end{aligned} \tag{2.16}$$

2.1.3 浮点算术运算程序设计的结果

具有有限十进制展开式的数可能有无穷的二进制展开式. 例如,

$$(0.1)_{10} = (0.000\ 110\ 011\ 001\ 100\ 110\ 011 \cdots)_2$$

所以, 不能按二进制浮点算术运算精确地表示 $(0.1)_{10}$. 在一台二进制机器上考察下列 MATLAB 代码, 就出现一个可能的问题:

```
x = 0.0
while x ~= 1.0
    x = x + 0.1
    disp([x,sqrt(x)])
end
```

这个代码形成一个无限的循环. 显然, 程序员希望当 $x = 1.0$ 时程序停止, 但是, 因为在计算机中不能精确表示十进制数 0.1, 所以决不能精确地得到数 1.0. 应该使用对舍入误差或截断误差不灵敏的循环试验.

在同时具有单精度和双精度的语言, 如 Fortran 和 C 中, 正确指定双精度常数是很重要的. 例如在 Fortran 中, 考虑语句

$$\mathrm{PI} = 3.141\,592\,653\,589\,79$$

假定 PI 为双精度变量. 即使右边的数已校正到双精度, 许多编译程序还是不能照此对它编码. 相反, 右边的数在编译阶段会被舍入成单精度常数, 然后在运行时, 再添上一些零把它扩展成双精度. 于是, 正确的做法是定义

$$\mathrm{PI} = 3.141\,592\,653\,589\,79\mathrm{D0}$$

这是双精度常数. 作为一个相关的例子, 考虑

$$\mathrm{PI} = 4.0^{*}\mathrm{ATAN}(1.0)$$

并声明 PI 为双精度变量. 当执行这个语句时, 先产生 π 的单精度近似, 再添加一些零把它扩展成双精度常数. 为了得到具有双精度的 π 的近似值, 可利用

$$\mathrm{PI} = 4.0\mathrm{D0}^{*}\mathrm{ATAN}(1.0\mathrm{D0})$$

又例如, 假定 H 为双精度, 考虑

$$\mathrm{H} = 0.1$$

对十进制数 0.1 这将产生只有单精度的 H 值. 正确的做法是用

$$\mathrm{H} = 0.1\mathrm{D0}$$

在 MATLAB 中, 所有的计算都是自动地按双精度来完成的. 我们不必注意特定双精度常数的格式. 换句话说, 对于 MATLAB, 我们可以更多地关注计算过程本身.

习题

1. 利用 MATLAB, 并且像 (2.6) 和 (2.7) 的例子中那样做, 求下列数字的二进制双精度 IEEE 浮点表示式.

(a) 8 **(b)** 12 **(c)** 1.5 **(d)** 0.5 **(e)** 1.25

2. 在以前使用的某些微型计算机上, 二进制浮点格式的指数 e 用 8 位, 符号 σ 用 1 位. 有效数字 $\bar{x}$ 包含 31 位, $1 \leqslant \bar{x} < 2$, 而且不把首位 1 隐藏起来. 算术运算也使用舍入. 为了确定表示式的精度, 求机器 ε[前面 (2.8) 所描述的] 和 (2.10) 的数 M. 并求出 (2.11) 的数 ε 的界.

3. 按你选择的语言编写一个程序实施下列算法 (以 MATLAB 给出). 在一台或多台你曾用过的计算机上运行它. 讨论你的结果.

```
  power = 1.0;
  b = 1.0;
  while b ~= 0.0
    power = power/2;
  a = 1.0 + power;
  b = a - 1.0;
  disp([power,a,b])
end
disp(['Unit round = ',num2str(2*power)])
```

4. 如果在一台使用截断的二进制计算机上运行下列程序, 输出是什么? 更确切地说, 估计程序员可能想要执行的这个循环的次数, 并叙述这个想要的状态是否为实际发生的情况. 如果语句 "$X = X + H$" 用 "$X = I * H$" 替代, 结果有何不同?

```
I = 0
X = 0.0
H = 0.1
WHILE X < 1.0
  I = I + 1
  X = X + H
  disp([I,X])
end
```

5. 下列 MATLAB 程序产生给定的输出. 解释结果.

```
  x = 0.0
  while x < 1.0
    x = x + 0.1;
  disp([x, sqrt(x)])
end
```

6. 设 $x > 0$ 满足 (2.2). 考虑使用 n 位精度的正二进制浮点数表示有效数字的计算机 [例如在 (2.2) 中 $n = 24$]. 假设把计算机外部的数 x 变成计算机内部的浮点逼近 $\text{fl}(x)$ 时使用

截断 (舍入归零).

(a) 证明

$$0 \leqslant x - \mathrm{fl}(x) \leqslant 2^{e-n+1}$$

(b) 证明 $x \geqslant 2^e$, 并用它证明

$$\frac{x - \mathrm{fl}(x)}{x} \leqslant 2^{-n+1}$$

(c) 设

$$\frac{x - \mathrm{fl}(x)}{x} = -\varepsilon$$

x	0.1	0.2	0.3	0.4	0.5	0.6
$\sqrt{x}$	0.316 2	0.447 2	0.547 7	0.632 5	0.707 1	0.774 6
x	0.7	0.8	0.9	1.0	1.1	
$\sqrt{x}$	0.836 7	0.894 4	0.948 7	1.000 0	1.048 8	

求解 $\mathrm{fl}(x)$. ε 的界是什么? (把这个结果推广到 $x < 0$ 的情况, 假设 $x > 0$ 只是为了简化代数运算.)

7. 设 $x > 0$ 满足 (2.2). 考虑使用 n 位精度的正二进制浮点数表示有效数字的计算机 [例如在 (2.2) 中 $n = 24$]. 假设把计算机外部的数 x 变成计算机内部的浮点逼近 $\mathrm{fl}(x)$ 时使用舍入.

(a) 证明

$$-2^{e-n} \leqslant x - \mathrm{fl}(x) \leqslant 2^{e-n}$$

(b) 证明 $x \geqslant 2^e$, 并用它证明

$$\frac{|x - \mathrm{fl}(x)|}{x} \leqslant 2^{-n}$$

(c) 设

$$\frac{x - \mathrm{fl}(x)}{x} = -\varepsilon$$

求解 $\mathrm{fl}(x)$. ε 的界是什么? (把这个结果推广到 $x < 0$ 的情况, 假设 $x > 0$ 只是为了简化代数运算.)

2.2 误差: 定义、来源和例题

定义计算量中的**误差**(error) 为

$$\text{误差} = \text{真值} - \text{近似值}$$

这样定义的量也称为**绝对误差**(absolute error). **相对误差**(relative error) 是相对于所求真值的误差大小的一个度量:

$$\text{相对误差} = \frac{\text{误差}}{\text{真值}}$$

此式给出误差与要逼近的真值之比的大小. 为了在用这些量计算时简化记号, 通常用 x_T 和 x_A 分别表示 x 的真值和近似值. 于是记

$$\text{Error}(x_A) = x_T - x_A$$

$$\text{Rel}(x_A) = \frac{x_T - x_A}{x_T}$$

例如, 考虑众所周知的逼近

$$\pi \doteq \frac{22}{7}$$

这里 $x_T = \pi = 3.141\ 592\ 65\cdots$, $x_A = 22/7 = 3.142\ 857\ 1\cdots$,

$$\text{Error}\left(\frac{22}{7}\right) = \pi - \frac{22}{7} \doteq -0.001\ 26$$

$$\text{Rel}\left(\frac{22}{7}\right) = \frac{\pi - (22/7)}{\pi} \doteq -0.000\ 402$$

另一个例子是泰勒余项 (1.11) 给出的误差.

相对误差更能刻划误差的本质. 例如, 假定两个城市之间的实际距离为 $d_T^{(1)} = 100\text{km}$, 而测量得到的距离是 $d_A^{(1)} = 99\text{km}$. 则

$$\text{Error}(d_A^{(1)}) = d_T^{(1)} - d_A^{(1)} = 1\text{km}$$

$$\text{Rel}(d_A^{(1)}) = \frac{\text{Error}(d_A^{(1)})}{d_T^{(1)}} = 0.01 \equiv 1\%$$

现在假定某两个商店之间的实际距离为 $d_T^{(2)} = 2\text{km}$, 它的估计量是 $d_A^{(2)} = 1\text{km}$, 则

$$\text{Error}(d_A^{(2)}) = d_T^{(2)} - d_A^{(2)} = 1\text{km}$$

$$\text{Rel}(d_A^{(2)}) = \frac{\text{Error}(d_A^{(2)})}{d_T^{(2)}} = 0.5 \equiv 50\%$$

两种情况中误差是相等的. 但与 $d_A^{(2)}$ 作为 $d_T^{(2)}$ 的估计相比, $d_A^{(1)}$ 显然是 $d_T^{(1)}$ 的更为精确的估计, 两个相对误差的大小反映了这个情况.

与相对误差有关的一个概念是**有效数字**(significant digits). 若数 x_A 开头的数字个数相对于真值 x_T 中相应的数字是正确的, 则称它为 x_A 的有效数字个数. 更精确的定义是, 假定数字用十进制形式写出, 计算误差 $|x_T - x_A|$ 的大小; 若这个误差小于或等于 x_T 的第 $m+1$ 位的 5 个单位, 则我们说, 相对于 x_T, x_A 从第 1 个非零数字起向右数至少有 m 位正确的有效数字.

例 2.2.1 (a) 相对于 $x_T = \frac{2}{9}$, $x_A = 0.222$ 有三位正确的数字.

(b) 相对于 $x_T = 23.494$, $x_A = 23.496$ 有四位正确的数字.

(c) 相对于 $x_T = 0.021\ 44$, $x_A = 0.021\ 38$ 只有两位正确的数字.

(d) 相对于 $x_T = \pi$, $x_A = \frac{22}{7}$ 有三位正确的数字. ■

可以证明, 若

$$\left|\frac{x_T - x_A}{x_T}\right| \leqslant 5 \cdot 10^{-m-1} \tag{2.17}$$

则相对于 x_T, x_A 有 m 位有效数字. 大部分人认为使用相对误差比有效数字方便, 所以一些教材利用 (2.17) 式来定义 x_A 有 m 位有效数字.

2.2.1 误差的来源

设想求解一个科学–数学问题, 并假定这个问题涉及一个计算过程. 这个过程通常包含几种不同类型的误差. 有时把误差分成 "原始的误差" 和 "结果的误差". 我们将给出可能出现的各种原始误差的一个粗略的分类.

(E1) 建模的误差 利用数学方程表示物理现实的过程称为*数学建模*(mathematical modeling). 描述实际问题的模型常常带有误差.

例如, 最简单的人口增长模型由下式给出

$$N(t) = N_0 \mathrm{e}^{kt} \tag{2.18}$$

其中 $N(t)$ 表示在时刻 t 时的人口, N_0 和 k 是正常数. 对于人口增长的某个阶段, 当资源无限制时, 这可能是一个精确的模型. 但当 t 很大时, 它往往会过高估计实际人口. 例如, 取 $k = 0.029\ 75$ 和 $N_0 = 3\ 929\ 000 \times \mathrm{e}^{-1\ 790k}$, 则从 $t = 1790$ 年到 $t = 1860$ 年这段时间内它精确地作为美国人口增长的模型, 但是它相当大地高估 1870 年的实际人口.

另一个例子出现在研究风疹麻疹的传播中. 在一定的假设之下, 有下列传染病在人群中传播的模型:

$$\frac{\mathrm{d}s(t)}{\mathrm{d}t} = -as(t)i(t)$$
$$\frac{\mathrm{d}i(t)}{\mathrm{d}t} = as(t)i(t) - bi(t)$$
$$\frac{\mathrm{d}r(t)}{\mathrm{d}t} = bi(t)$$

其中 $s(t), i(t)$ 和 $r(t)$ 分别表示时刻 t 时人群中*易受感染的*(susceptible) 人, *受传染的*(infectious) 人, 以及*排除*(removed) 上述两种人之外的人的比例. 所有的变量是时间 t 的函数. 常数可以取

$$a = \frac{6.8}{11}, \quad b = \frac{1}{11}$$

对某些其他的疾病 (如流感), 这个模型同样适用, 但是需要适当改变常数 a 和常数 b. 这个模型又是现实的一个有用的逼近.

数学模型中误差的研究超出数值分析的范围, 但这种误差是求解整个科学问题的误差之一.

(E2) 计算过程中的误差 几乎每个人都熟悉这些误差. 在计算机发明以前, 这种错误一般只是孤立的算术误差, 需要用复杂的检查方案去发现它们. 今天, 这类错误更有可能是程序设计不当造成的. 为了发现这些差错, 拥有某种检验程序输出准确性的方法是重要的. 当第一次运行程序时, 利用已知正确答案的案例. 对于一个复杂的程序, 把它拆成几个小的子程序, 每个子程序可以分别地作试验. 当确信整个程序是正确的并对关注的案例运行时, 要始终警惕输出是否合理.

(E3) 物理测量误差 许多问题涉及物理数据, 而这些数据包含观测误差. 例如, 在真空中光速是

$$c = (2.997\,925 + \varepsilon) \cdot 10^{10}\mathrm{cm/s}, \quad |\varepsilon| \leqslant 0.000\,003 \tag{2.19}$$

因为物理数据包含误差, 所以基于这个数据的计算将包含这个观测误差的影响. 数值分析不能去掉数据中的误差, 但它可以考虑计算中误差传播的影响. 数值分析还能够建议计算的最好形式, 它能使数据中误差传播的影响达到最小. 2.3 节讨论考虑算术运算计算中误差的传播.

(E4) 机器表示和算术运算误差 这些误差出现在使用计算机或计算器时, 例如 2.1 节中讨论的舍入误差或截断误差. 当使用浮点算术运算时, 这些误差是不可避免的; 在一些问题中它们构成主要的误差来源. 2.4 节中, 我们将对某些求和过程分析舍入误差的影响.

(E5) 数学上的逼近误差 数学上的逼近误差是后面各章关注的主要的误差形式. 为了说明这类误差, 考虑求下列积分值

$$I = \int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x$$

e^{-x^2} 没有能用初等函数表示的不定积分, 因此上述积分不能精确地求值. 于是用一个可以计算的量来逼近积分. 例如, 利用泰勒逼近

$$\mathrm{e}^{-x^2} \approx 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!}$$

则

$$I \approx \int_0^1 \left(1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!}\right)\mathrm{d}x \tag{2.20}$$

上式可以容易地求值. (2.20) 中的误差称为*数学逼近误差*(mathematical approximation error), 有些人称它为*截断误差*(truncation error) 或*离散误差*(discretization error). 当给定的问题不能精确地求解而用可以求解的问题逼近时就会出现这样的误差. 我们会设法估计这些数学逼近误差的大小并使它们充分小. 上面的近似积分 (2.20) 的误差可使用前面的泰勒公式 (1.18) 来研究.

现在将描述在实践中普遍出现的三种重要的误差. 它们有时被认为是误差的来源, 但我们把它们看成是从上面描述的来源 (E1) 到 (E5) 导出的. 通常可以通过选择计算的形式使这些误差达到极小.

2.2.2　有效数字损失的误差

有效数字损失(loss of significant digits) 的概念最好是通过考察一些具体例子来理解. 首先考虑, 对 x 值的一个递增序列, 求

$$f(x) = x[\sqrt{x+1} - \sqrt{x}\,] \tag{2.21}$$

的值. 在表 2-6 中给出利用 6 位十进制计算器的结果. 随着 x 值的增大, $f(x)$ 的计算值的有效数字的位数减少.

表 2-6　(2.21) 的值

x	计算的 $f(x)$	真实的 $f(x)$
1	0.414 210	0.414 214
10	1.543 40	1.543 47
100	4.990 00	4.987 56
1000	15.800 0	15.807 4
10.000	50.000 0	49.998 8
100.000	100.000	158.113

为更好地理解这种情况, 我们来看当 $x = 100$ 时计算的各个步骤. 在计算器上,

$$\sqrt{100} = 10.000\,0, \quad \sqrt{101} = 10.049\,9$$

第一个值是精确的, 第二个值有 6 位有效数字. 下一步,

$$\sqrt{x+1} - \sqrt{x} = \sqrt{101} - \sqrt{100} = 0.049\,900\,0 \tag{2.22}$$

然而真实的值应该是 0.049 875 6. 计算 (2.22) 存在*有效数字的损失*. 在 $\sqrt{x+1} = \sqrt{101}$ 中 3 位精确的数字通过减去 $\sqrt{x} = \sqrt{100}$ 中对应的数字被消去. 精确度的损失是 $f(x)$ 的形式以及使用有限精度 6 位十进制算术的副产品.

对这个具体的 $f(x)$, 可以通过一个简单的方法避免有效数字的损失. 把 (2.21) 看作分母为 1 的分式, 并用 $\sqrt{x+1}+\sqrt{x}$ 乘以分子和分母, 得到

$$f(x)=x\frac{\sqrt{x+1}-\sqrt{x}}{1}\cdot\frac{\sqrt{x+1}+\sqrt{x}}{\sqrt{x+1}+\sqrt{x}}=\frac{x}{\sqrt{x+1}+\sqrt{x}} \tag{2.23}$$

后面的表达式在求值中不会导致有效数字的损失. 在 6 位十进制计算器上, (2.23) 给出

$$f(100)=4.987\ 56$$

答案正确到 6 位数字.

第二个例子, 对于趋近于 0 的 x 值序列, 求

$$f(x)=\frac{1-\cos(x)}{x^2} \tag{2.24}$$

的值. 在表 2-7 中给出利用流行的 10 位十进制计算器计算的结果. 为了解精度的损失过程, 我们查看 $x=0.01$ 时计算的各个步骤. 首先, 在计算器上

$$\cos(0.01)=0.999\ 950\ 000\ 4$$

这个结果有 9 位精确的有效数字, 它的第 10 位数字偏离 2 个单位. 下一步, 计算

$$1-\cos(0.01)=0.000\ 049\ 999\ 6$$

这个结果只有 5 位有效数字, 因为有 4 位数字在减法中丢失. 再除以 $x^2=0.000\ 1$ 就得到表中记载的值.

表 2-7 在 10 位十进制计算器上 (2.24) 的值

x	计算的 $f(x)$	真实的 $f(x)$
0.1	0.499 583 470 0	0.499 583 472 2
0.01	0.499 996 000 0	0.499 995 833 3
0.001	0.500 000 000 0	0.499 999 958 3
0.000 1	0.500 000 000 0	0.499 999 999 6
0.000 01	0.0	0.500 000 000 0

为避免有效数字损失, 我们可用另一个公式来表示 $f(x)$, 这个公式可以避免几乎相等的值之间的减法. 利用 (1.15) 中给出的泰勒逼近, 我们得到

$$\begin{aligned}\cos(x)=&1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+R_6(x)\\ R_6(x)=&\frac{x^8}{8!}\cos(\xi)\end{aligned}$$

其中 ξ 是 0 和 x 之间的一个未知数. 所以

$$\begin{aligned}f(x) &=\frac{1}{x^2}\left\{1-\left[1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+R_6(x)\right]\right\}\\&=\frac{1}{2!}-\frac{x^2}{4!}+\frac{x^4}{6!}-\frac{x^6}{8!}\cos(\xi)\end{aligned}$$

于是, $f(0)=\frac{1}{2}$. 对于 $|x|\leqslant 0.1$,

$$\left|\frac{x^6}{8!}\cos(\xi)\right|\leqslant\frac{10^{-6}}{8!}\doteq 2.5\times 10^{-11} \tag{2.25}$$

因此,

$$f(x)\approx\frac{1}{2!}-\frac{x^2}{4!}+\frac{x^4}{6!},\quad |x|\leqslant 0.1$$

具有 (2.25) 给出的精度. 对小的 x 值, 这里给出求 $f(x)$ 值的一个较好的方法.

当两个几乎相等的值相减时, 开头的有效数字会丢失. 有时这是显然的, 如上面的例子 (2.21) 和例子 (2.24), 于是通常可以找到避免有效数字损失的方法. 而在另一些情形, 有效数字的损失不容易发现. 一种常见的情况出现在计算包含若干项的和中, 例如在利用泰勒多项式逼近函数 $f(x)$ 时. 若和的值相对于求和的某些项比较小的话, 则有可能在求和过程中失去某些有效数字的精度.

作为上述现象的一个例子, 考察用 e^x 的泰勒级数逼近 (1.13) 计算 e^{-5}:

$$\mathrm{e}^{-5}=1+\frac{(-5)}{1!}+\frac{(-5)^2}{2!}+\frac{(-5)^3}{3!}+\frac{(-5)^4}{4!}+\cdots \tag{2.26}$$

假定使用具有 4 位十进制浮点算术运算的计算机, 则这个级数的每一项必须舍入到 4 位有效数字. 在表 2-8 中, 我们给出这些项以及这些项的数值和. e^{-5} 的 4 位有效数字的真实值是 0.006 738, 这与表中最后的和大不相同. 只有当用更高精度计算 (2.26) 中次数 ⩽25 的项, 方能得到 e^{-5} 的精确到 4 位有效数字的近似值.

在这个例子中, 有些项相对较大, 但是它们的和却是一个较小的数 e^{-5}. 这意味着求和计算中存在有效数字损失的误差. 3 次项中的舍入误差与表中最后答案的误差具有相同的数量级. 在这个例子中, 要避免有效数字损失是十分容易的, 可利用

$$\mathrm{e}^{-5}=\frac{1}{\mathrm{e}^5}$$

计算 e^5 的级数不涉及正项和负项相互抵消, 或者更简单地, 先计算 $\mathrm{e}^{-1}=1/\mathrm{e}$, 再用它本身自乘 4 次形成 e^{-5}. 对于大多数其他的级数, 通常不存在如此简单的解决方法.

一般准则

假如 S 由一连串数相加得到. 若 S 的值比求和的某些基小得多, 则 S 有可能包含有效数字损失的误差. (2.27)

表 2-8 利用四位十进制算术运算计算 (2.26)

次数	项	和	次数	项	和
0	1.000	1.000	13	−0.196 0	−0.042 30
1	−5.000	−4.000	14	0.700 1E−1	0.027 71
2	12.50	8.500	15	−0.233 4E−1	0.004 370
3	−20.83	−12.33	16	0.729 3E−2	0.011 66
4	26.04	13.71	17	−0.214 5E−2	0.009 518
5	−26.04	−12.33	18	0.595 8E−3	0.010 11
6	21.70	9.370	19	−0.156 8E−3	0.009 957
7	−15.50	−6.130	20	0.392 0E−4	0.009 996
8	9.688	3.558	21	−0.933 3E−5	0.009 987
9	−5.382	−1.824	22	0.212 1E−5	0.009 989
10	2.691	0.867 0	23	−0.461 1E−6	0.009 989
11	−1.223	−0.356 0	24	0.960 7E−7	0.009 989
12	0.509 7	0.153 7	25	−0.192 1E−7	0.009 989

2.2.3 函数求值中的噪声

考虑对区间 $a \leqslant x \leqslant b$ 中的一切 x 求函数 $f(x)$ 的值. 若函数连续, 则这个函数的图形是连续曲线. 下面, 考虑在一台使用舍入或截断浮点算术运算的计算机上计算 $f(x)$ 的值. $f(x)$ 求值中算术运算 (例如加法和乘法) 引起的误差一般是十分小的. 若非常仔细地观察 $f(x)$ 计算值的图形, 它将不再是一条连续的曲线, 而是一条反映求值过程中误差的 "模糊" 曲线.

例 2.2.2 求 $f(x) = (x-1)^3$ 的值. 利用 MATLAB 并用嵌套形式

$$f(x) = -1 + x(3 + x(-3 + x)) \tag{2.28}$$

来计算. MATLAB 中的算术运算使用 IEEE 双精度数和标准舍入. 图 2-1 给出 $0 \leqslant x \leqslant 2$ 上 $f(x)$ 计算值的图形, 它看上去是一条光滑的连续曲线. 下面研究这条曲线的一小段, 取 $0.999\,98 \leqslant x \leqslant 1.000\,02$, 对 $[0.999\,98, 1.000\,02]$ 中 x 的 81 个等距值, 图 2-2 中给出 $f(x)$ 计算值的详细图. 注意, $f(x)$ 的图形似乎不像是取自连续曲线, 倒像是一条狭窄的随机取值的 "模糊带". 对于 $f(x)$ 计算曲线的所有部分都是这样的, 但是仅当非常仔细地查看曲线时它才变得明显. ■

应该把计算机上的函数 $f(x)$ 看作随机取值的模糊带, 在大多数情况下模糊带的垂直带宽十分小. 这通常不会引起严重的问题, 但在有些场合, 我们得十分注意. 例如, 考虑 (2.28) 的求根, 由于有许多符号变更, 图 2-2 似乎表明 $f(x)$ 在 $[0.999\,98, 1.000\,02]$ 中有大量的根.

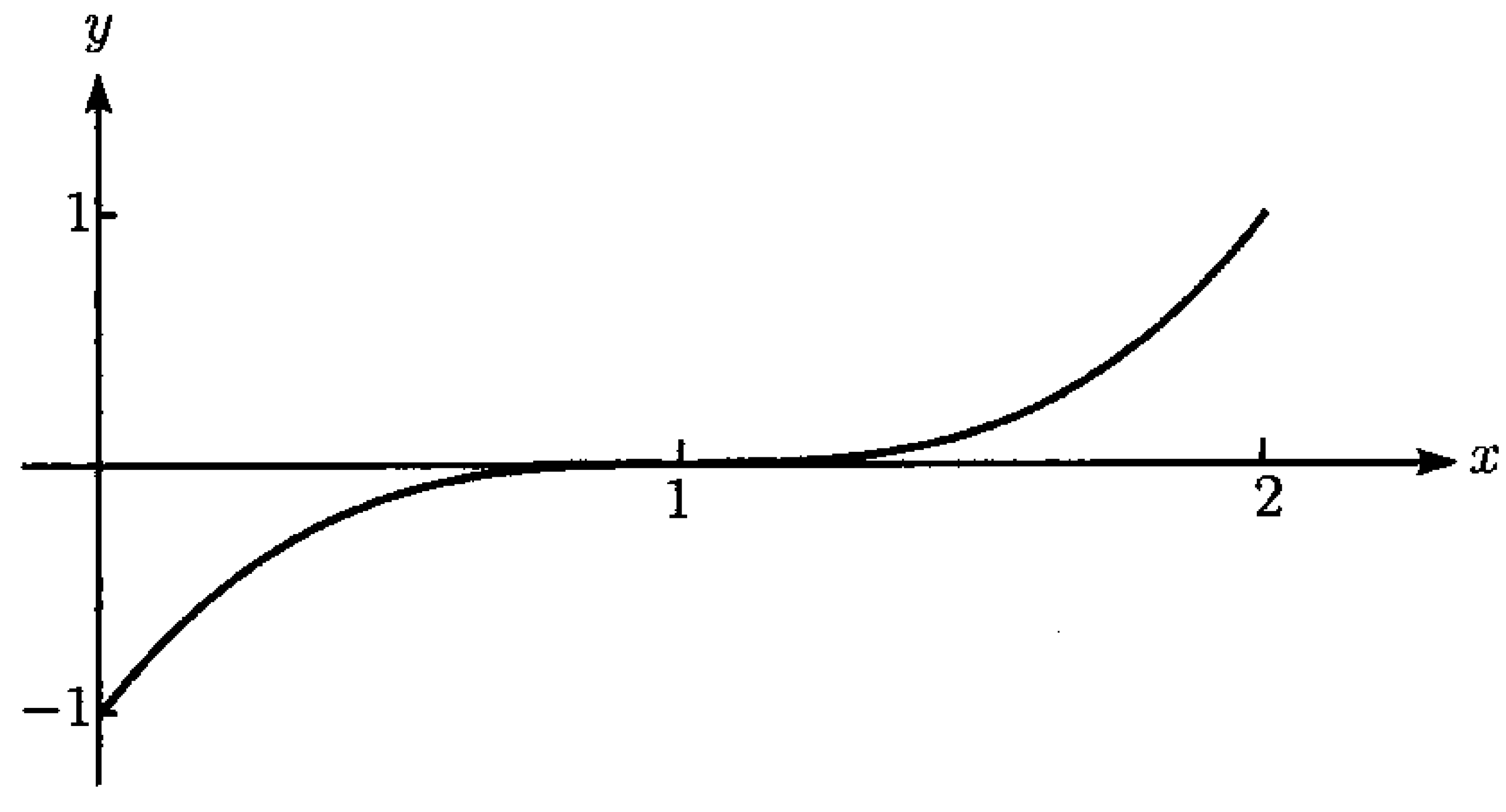


图 2-1 $f(x)=x^3-3x^2+3x-1$

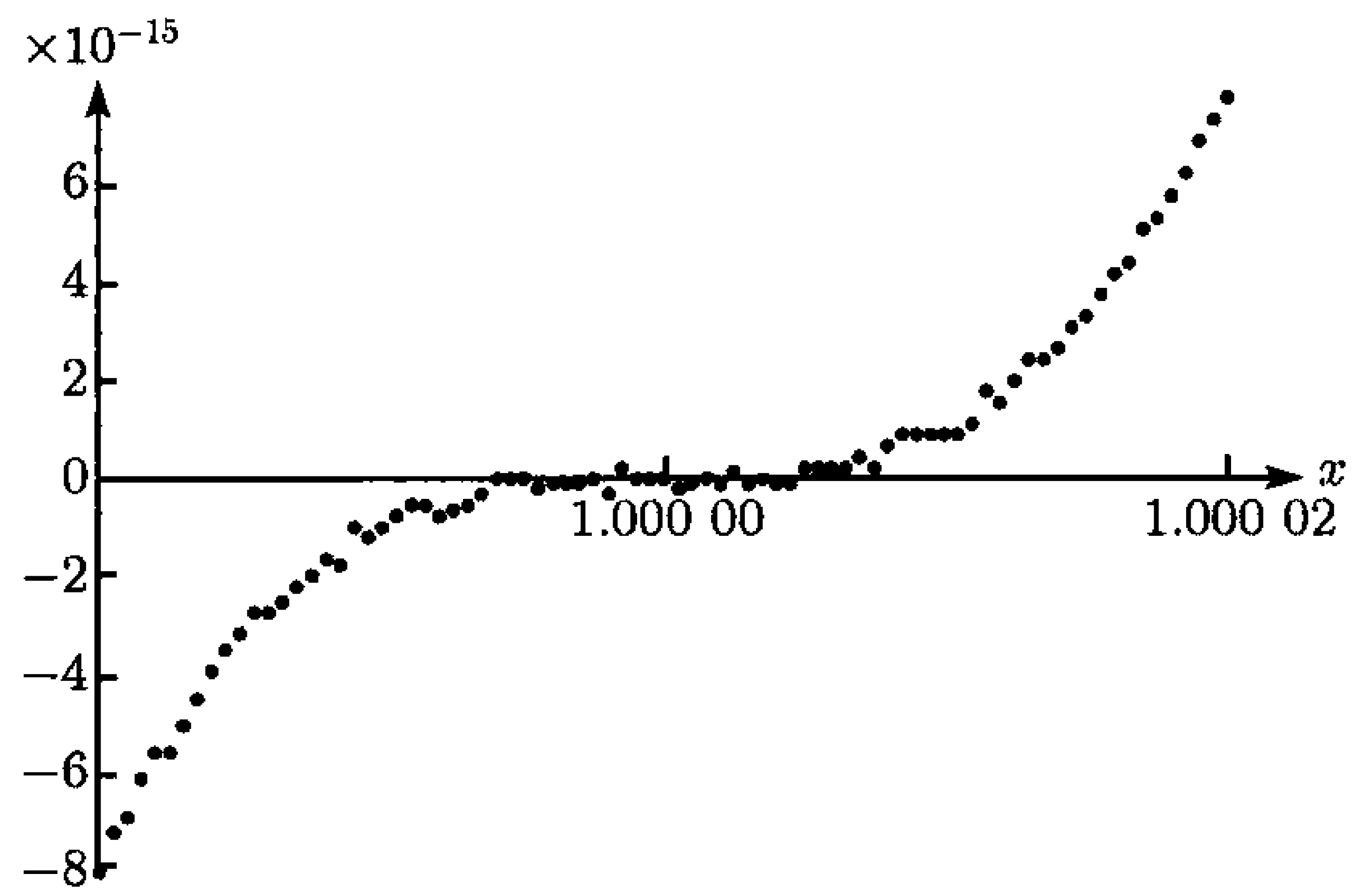


图 2-2 $f(x)=x^3-3x^2+3x-1$ 在 x=1 附近的一个详细的图形

2.2.4 下溢误差和上溢误差

根据 2.1 节中给出的浮点数定义, 能按浮点形式表达的数的值存在上限和下限. 企图产生一些太小的数将导致**下溢**(underflow) 误差. 大部分计算机上规定这样的数为零然后继续计算.

例如, 考虑对接近于零的 x, 求

$$f(x)=x^{10} \tag{2.29}$$

的值. 当利用 IEEE 单精度算术运算时, 在**正规化**(normalized) 浮点格式中能表达的最小非零的正数是

$$m=2^{-126}\doteq 1.18\times 10^{-38} \tag{2.30}$$

见表 2-2 中 $E=1$ 和 $(a_1a_2\cdots a_{23})=(00\cdots 0)_2$ 时的数. 回顾 (2.4) 提及的**非正规化**(unnormalized) 浮点单精度格式和表 2-2, 其中 $E=0$. 则当

$$x^{10}<m$$

$$|x| < \sqrt[10]{m} \doteq 1.61 \times 10^{-4}$$

$$-0.000\ 161 < x < 0.000\ 161$$

时 $f(x)$ 为零. 对其他浮点格式有类似的结果, 只是实际的界依赖于浮点表示式的指数范围. 某些程序也允许使用非正规化浮点数 (例如 MATLAB), 这些数的绝对值较小, 而且有效数字的位数也少.

使用一些对浮点格式而言太大的数将导致上溢(overflow) 误差, 在大部分计算机上这些误差一般是致命的. 对于 IEEE 浮点格式, 上溢误差可能包含 $\pm\infty$ 或 NaN 的值, 它和上下文有关. 通常, 上溢误差暗示程序中一个更重要的问题或误差, 用户有必要察觉这样的误差.

在少数情况, 有可能通过改写求值的表达式来消除上溢误差. 例如, 考虑求

$$z = \sqrt{x^2 + y^2}$$

的值. 若 x 或 y 非常大, 则 $x^2 + y^2$ 可能产生一个上溢误差, 即使 z 仍可能在机器的浮点范围中. 为避免这种情况, 可以用等价的公式

$$z = \begin{cases} |x|\sqrt{1 + (y/x)^2}, & 0 \leqslant |y| \leqslant |x| \\ |y|\sqrt{1 + (x/y)^2}, & 0 \leqslant |x| \leqslant |y| \end{cases}$$

两种情况中平方根的自变量具有形式 $1 + \omega^2$, $|\omega| \leqslant 1$. 这个计算不会产生任何上溢误差, 除非所使用的浮点格式表示的 z 太大.

习题

1. 计算下列逼近 $x_A \approx x_T$ 的误差、相对误差和有效数字位数:

(a) $x_T = 28.254, x_A = 28.271$　　(b) $x_T = 0.028\ 254, x_A = 0.028\ 271$

(c) $x_T = \mathrm{e}, x_A = 19/7$　　(d) $x_T = \sqrt{2}, x_A = 1.414$

(e) $x_T = \ln(2), x_A = 0.7$

2. (a) 在 (2.18) 的人口增长模型中, $N(t) = N_0\mathrm{e}^{kt}$, 给出常数 N_0 的物理意义.

(b) 证明 $N(t)$ 满足关系

$$\frac{N(t+1)}{N(t)} = \text{常数}$$

因而, $N(t+1) =$(常数)$\cdot N(t)$, 并且 t 每增加 1 个单位, 人口以常数比率增长. 依自然法则这是合理的吗?

3. 一个稍微更加复杂的人口增长模型是对某些正常数 N_c 和 b,

$$N(t) = \frac{N_c}{1 + \mathrm{e}^{-bt}}, t \geqslant 0$$

找出这个模型和习题 2 中的模型之间的主要差别. 常数 N_c 的物理意义是什么?
提示: 查看当 t 变大时每个模型的性态.

4. 利用所使用的泰勒多项式的余项公式界定 (2.20) 中的误差.

5. 在某些情况, 通过重新安排求值的函数可以避免有效数字损失的误差, 如 (2.23) 中对 $f(x)$ 所做的那样. 对下列情况做类似处理, 某些情况利用三角恒等式. 除了 (b) 以外的所有情况, 假定 x 接近于 0:

(a) $\dfrac{1-\cos(x)}{x^2}$　　(b) $\ln(x+1)-\ln(x), x$ 大

(c) $\sin(a+x)-\sin(a)$　　(d) $\sqrt[3]{1+x}-1$

(e) $\dfrac{\sqrt{4+x}-2}{x}$

6. 当 x 接近于 0 时, 利用泰勒多项式逼近来避免下列公式中有效数字的损失:

(a) $\dfrac{e^x-1}{x}$　　(b) $\dfrac{1-e^{-x}}{x}$　　(c) $\dfrac{e^x-e^{-x}}{2x}$

(d) $\dfrac{\ln(1-x)+xe^{x/2}}{x^3}$　　(e) $\dfrac{1-(1-x)^{\sqrt{2}-1}}{x}$　　(f) $\dfrac{x-\sin(x)}{x^3}$

(g) $\dfrac{x-\sin(x)}{\tan(x)}$　　(h) $\dfrac{x+\ln(1-x)}{x^2}$

7. 考虑恒等式

$$\int_0^x \sin(xt)\mathrm{d}t=\frac{1-\cos(x^2)}{x}$$

当 x 接近于 0 时, 说明利用右端的分式求表达式值的困难, 给出一个避免这个问题并使结果尽可能精确的方法.

8. 对恒等式

$$f(x)=\frac{1}{x}\int_0^x e^{-xt}\mathrm{d}t=\frac{1-e^{-x^2}}{x^2}, x\neq 0$$

重复习题 7.

9. (a) 利用二次公式解方程 $x^2-26x+1=0$. 利用 5 位十进制数算术运算求方程根的数值; 例如, 你将需要 $\sqrt{168}\doteq 12.961$. 识别你遇到的任何有效数字损失的误差.

(b) 只用 5 位十进制数算术运算精确地求出两个根.

提示: 利用

$$13-\sqrt{168}=\frac{1}{13+\sqrt{168}}$$

10. 对方程 $x^2-40x+1=0$ 重复习题 9, 利用 $\sqrt{399}\doteq 19.975$.

11. 讨论求解二次方程 $ax^2+bx+c=0$ 中可能遇到的有效数字损失的误差. 如何避免有效数字损失的误差?

12. 对接近于 0 的 x 值, 数值上考察恒等式

$$\sin(x)=\sqrt{1-\cos^2(x)},\quad 0\leqslant x\leqslant\frac{\pi}{2}$$

的精度. 取 x 靠近于 0 的值并考察 $\sqrt{1-\cos^2(x)}$ 的计算值中的相对误差. 利用直接计算的 $\sin(x)$ 的值对它们作比较.

13. 对大的 x 值, 求

$$f(x)=\sqrt{1+\frac{1}{x}}-1$$

的精确值. 计算

$$\lim_{x\to\infty} xf(x)$$

14. 考虑对大的 x 利用泰勒逼近

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \cdots + (-1)^n \frac{x^{2n}}{(2n)!}$$

求 $\cos(x)$ 的值. 为看出使用这个逼近的困难, 利用它计算 $\cos(2\pi) = 1$:

$$\cos(2\pi) \approx 1 - \frac{(2\pi)^2}{2!} + \frac{(2\pi)^4}{4!} - \cdots + (-1)^n \frac{(2\pi)^{2n}}{(2n)!}$$

假设使用十进制数算术运算, 比如说, 带舍入的四位十进制浮点算术运算. 若 $2n = 20$, 则在这个逼近中的误差是 0.000 32, 因而, 相对于使用浮点算术运算, 多项式应该是足够精确的. 现在计算每一项并把它舍入到 4 位有效数字, 求出它的精确的浮点表示. 例如, 前 3 项是 1.000, −19.74, 64.94. 求出这 11 项, 把它们精确地相加. 试问它们逼近 $\cos(2\pi) = 1.000$ 的密切程度如何? 说明不精确的原因.

15. 对函数 $f(x) = (x-1)^3$ 重复例 2.2.2, 但利用

$$f(x) = x^3 - 3x^2 + 3x - 1$$

而不是 (2.28) 的嵌套形式. 注意任何可能出现的差别.

16. 在给定的区间上计算下列多项式 $p(x)$ 的值. 以给定的形式和嵌套形式分别计算 $p(x)$ 的值. 在给定的区间上使用相当多的点 x. 两组函数值集合都将包含 "噪声", 但一般地它们是不同的, 因为对 $p(x)$ 使用不同的公式. 在给定的区间上, 从对应的值直接减去嵌套值, 并打印这些数. 这给出了计算 $p(x)$ 的两种方法的噪声的合成, 它能让我们大致了解噪声的大小. 分别画出 $p(x)$ 的图形和合成噪声的图形.

(a) $x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.795\ 1, \quad 0.9 \leqslant x \leqslant 1.3$

(b) $x^5 + 0.9x^4 - 1.62x^3 - 1.458x^2 + 0.656\ 1x + 0.590\ 49, \quad -1.1 \leqslant x \leqslant -0.6$

17. 利用多项式

$$p(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.795\ 1$$

重复例题 2.2.2 和习题 15. 仔细观察当 x 接近 1.1 时 $p(x)$ 的值.

18. 为在你的计算机上生成一个上溢误差, 编写一个程序, 对数 $x > 1$ 反复地平方并打印结果. 最终, 你的数字会超出机器的浮点数指数界限.

19. 利用 IEEE 双精度正规化浮点算术运算使 x^{10} 下溢的 x 值是什么? 什么时候它上溢?

2.3 误差的传播

如果计算中用到包含误差的数, 则计算结果会受到这些误差影响. 首先, 我们讨论用这类数对普通算术运算的影响, 然后考察对更一般的函数求值的影响. 设 x_A 和 y_A 表示计算中使用的数, 而设 x_T 和 y_T 是对应的真值. 我们希望界定

$$E = (x_T \omega y_T) - (x_A \omega y_A) \tag{2.31}$$

其中ω 表示运算 "+", "−", "·" 或 "÷" 之一. 误差 E 称为**传播误差**(propagated error).

用于界定 E 的第一种方法通称**区间算术运算**(interval arithmetic). 假定知道 $x_T - x_A$ 和 $y_T - y_A$ 的界. 然后利用这些界以及 $x_A \omega y_A$, 寻找一个保证包含 $x_T \omega y_T$ 的区间.

例 2.3.1 设 $x_A = 3.14$ 和 $y_A = 2.651$ 是由 x_T 和 y_T 正确地舍入到所得到的数字. 则

$$|x_T - x_A| \leqslant 0.005, \quad |y_T - y_A| \leqslant 0.000\,5$$

或等价地

$$3.135 \leqslant x_T \leqslant 3.145, \quad 2.650\,5 \leqslant y_T \leqslant 2.651\,5 \tag{2.32}$$

对加法运算

$$x_A + y_A = 5.791 \tag{2.33}$$

对于真值, 利用 (2.32) 得到界定的区间

$$\begin{aligned} 3.135 + 2.650\,5 \leqslant x_T + y_T \leqslant 3.145 + 2.651\,5 \\ 5.785\,5 \leqslant x_T + y_T \leqslant 5.796\,5 \end{aligned} \tag{2.34}$$

为了得到传播误差的界, 从 (2.34) 中减去 (2.33) 得到

$$-0.005\,5 \leqslant (x_T + y_T) - (x_A + y_A) \leqslant 0.005\,5$$

作除法

$$\frac{x_A}{y_A} = \frac{3.14}{2.651} \doteq 1.184\,459 \tag{2.35}$$

而且

$$\frac{3.135}{2.651\,5} \leqslant \frac{x_T}{y_T} \leqslant \frac{3.145}{2.650\,5}$$

分数相除并舍入到 7 位数字, 得到

$$1.182\,350 \leqslant \frac{x_T}{y_T} \leqslant 1.186\,569 \tag{2.36}$$

对于误差而言,

$$-0.002\,109 \leqslant \frac{x_T}{y_T} - 1.184\,459 \leqslant 0.002\,110$$ ■

这种方法保证得到含有精确答案所在的区间, 称为**区间算术运算**. 它是一种有用的方法, 而且曾在计算机上同时用软件和硬件实施过这种运算. 但是对于连续的计算, 必须很小心地操作区间算术运算, 否则它将导致预报的误差界远远超过真正的误差. 在开发区间分析的实际操作方面已有很大的进展, 但是还未得到一种广

泛地使用的控制误差或界定实际计算中误差的技巧. 将来这种情况似乎有可能改变.

乘法中的传播误差(propagated error in multiplication) 我们来比较 x_Ty_T 和 x_Ay_A. x_Ay_A 相对于 x_Ty_T 的相对误差是

$$\text{Rel}(x_Ay_A) = \frac{x_Ty_T - x_Ay_A}{x_Ty_T}$$

设 $x_T = x_A + \varepsilon$, $y_T = y_A + \eta$. 则

$$\begin{aligned}\text{Rel}(x_Ay_A) &= \frac{x_Ty_T - (x_T - \varepsilon)(y_T - \eta)}{x_Ty_T} \\ &= \frac{\eta x_T + \varepsilon y_T - \varepsilon\eta}{x_Ty_T} \\ &= \frac{\varepsilon}{x_T} + \frac{\eta}{y_T} - \left(\frac{\varepsilon}{x_T}\right)\left(\frac{\eta}{y_T}\right) \\ &= \text{Rel}(x_A) + \text{Rel}(y_A) - \text{Rel}(x_A)\text{Rel}(y_A)\end{aligned} \tag{2.37}$$

当 $\text{Rel}(x_A)$ 和 $\text{Rel}(y_A)$ 的误差比 1 小很多时,

$$\text{Rel}(x_Ay_A) \approx \text{Rel}(x_A) + \text{Rel}(y_A) \tag{2.38}$$

上式表明乘法中的相对误差传播缓慢, 这是一个令人满意的性质. 使用除法时的传播误差在习题 9 中考察, 它具有类似的良好性质.

加法和减法中的传播误差 加法和减法运算不一定是良态的. 有可能 $\text{Rel}(x_A)$ 和 $\text{Rel}(y_A)$ 的值较小, 而 $\text{Rel}(x_A \pm y_A)$ 比较大; 2.2 节的习题 9 和习题 10 就是这种情况. 在习题 9 中, 设

$$x_T = x_A = 13$$

$$y_T = \sqrt{168}, \quad y_A = 12.961$$

关于相对误差

$$\text{Rel}(x_A) = 0, \quad \text{Rel}(y_A) \doteq 0.000\,037\,1$$

$$\text{Error}(x_A - y_A) \doteq -0.000\,481\,4$$

$$\text{Rel}(x_A - y_A) \doteq -0.012\,5$$

这个误差的来源与有效数字损失的误差密切相关.

总的计算误差 当在计算机上使用浮点算术运算时, $x_A\omega y_A$ 的计算包含附加的舍入误差或截断误差, 如 (2.35) 式中那样. $x_A\omega y_A$ 的计算值包含传播误差以及一个舍

入误差或截断误差. 更确切地讲, 设 $\hat{\omega}$ 表示在计算机上执行的完整的计算, 包括任意的舍入或截断. 则总的误差为

$$(x_T\omega y_T)-(x_A\hat{\omega}y_A)=[(x_T\omega y_T)-(x_A\omega y_A)]+[(x_A\omega y_A)-(x_A\hat{\omega}y_A)] \tag{2.39}$$

右边的第一项是传播误差, 第二项是计算 $x_A\omega y_A$ 时产生的误差.

当对基本的算术运算使用 IEEE 算术运算时, 有

$$x_A\hat{\omega}y_A=\mathrm{fl}(x_A\omega y_A) \tag{2.40}$$

也就是说先精确地计算 $x_A\omega y_A$, 然后舍入或截断到标准的浮点长度. 关于误差, 利用 (2.11), 记

$$\mathrm{fl}(x_A\omega y_A)=(1+\varepsilon)(x_A\omega y_A)$$

其中 ε 如同在 (2.12)~(2.16) 中那样界定. 把此式与 (2.40) 联合起来可得

$$\begin{aligned}(x_A\omega y_A)-(x_A\hat{\omega}y_A)&=-\varepsilon(x_A\omega y_A)\\ \frac{(x_A\omega y_A)-(x_A\hat{\omega}y_A)}{x_A\omega y_A}&=-\varepsilon\end{aligned} \tag{2.41}$$

因而, 与 $x_A\omega y_A$ 相比, 舍入或截断过程把一个相对小的新误差引入到 $x_A\hat{\omega}y_A$ 中.

函数求值中的传播误差

求 $f(x)$ 在近似值 x_A 处的值而不是在 x_T 处的值, 需要考虑 $f(x_A)$ 是否接近 $f(x_T)$? 在 $f(x_A)$ 的实际求值中也会引入一个附加的误差, 这是 2.2 节中描述过的噪声现象. 但在此我们只比较 $f(x_A)$ 的精确值与 $f(x_T)$ 的精确值.

假如对 $a\leqslant x\leqslant b$, $f'(x)$ 存在且连续. 利用中值定理 (附录 A 的定理 A.4, 公式 A.3), 得到

$$f(x_T)-f(x_A)=f'(c)(x_T-x_A) \tag{2.42}$$

其中 c 是 x_A 和 x_T 之间的一个未知点. 因为通常 x_A 和 x_T 很接近, 所以

$$f(x_T)-f(x_A)\approx f'(x_T)(x_T-x_A)\approx f'(x_A)(x_T-x_A) \tag{2.43}$$

在大多数场合, 最好考虑相对误差

$$\mathrm{Rel}(f(x_A))\approx\frac{f'(x_T)}{f(x_T)}(x_T-x_A)=\frac{f'(x_T)}{f(x_T)}x_T\mathrm{Rel}(x_A) \tag{2.44}$$

例 2.3.2　在化学中, 人们研究理想气体定律

$$PV=nRT$$

其中对任何气体 R 都是一个常数. 在 MKS(米–千克–秒) 度量系统中,

$$R = 8.314\,3 + \varepsilon, \quad |\varepsilon| \leqslant 0.001\,2$$

为了明确 R 中这个不确定量对计算 T 的影响, 考虑求 T 的值, 假定 $P = V = n = 1$. 则

$$T = \frac{1}{R}$$

定义

$$f(x) = \frac{1}{x}$$

并在 $x = R$ 处求它的值. 设

$$x_T = R, \quad x_A = 8.314\,3$$

假定

$$|x_T - x_A| \leqslant 0.001\,2$$

关于误差

$$E = \frac{1}{R} - \frac{1}{8.314\,3} \tag{2.45}$$

从 (2.43) 有

$$|E| = |f(x_T) - f(x_A)| \approx |f'(x_A)|\,|x_T - x_A| \leqslant \left(\frac{1}{x_A^2}\right)(0.001\,2) \doteq 1.74 \times 10^{-5}$$

关于相对误差

$$\left|\frac{E}{f(x_T)}\right| = \left|\frac{f(x_T) - f(x_A)}{f(x_T)}\right| \leqslant \frac{1.74 \times 10^{-5}}{0.120\,2} \doteq 0.000\,144$$

因而, R 中的不确定量在计算值

$$\frac{1}{R} \doteq \frac{1}{8.314\,3}$$

中导致一个相对较小的误差. ■

例 2.3.3 应用 (2.42) 求

$$f(x) = b^x$$

的值, 其中 b 是一个正常数. 利用

$$f'(x) = (\ln b)b^x$$

从公式 (2.43) 和 (2.44) 得到

$$\begin{aligned} b^{x_T} - b^{x_A} &\approx (\ln b) b^{x_T}(x_T - x_A) \\ \text{Rel}(b^{x_A}) &\approx (\ln b) x_T \text{Rel}(x_A) \end{aligned} \tag{2.46}$$

若量

$$K = (\ln b) x_T \tag{2.47}$$

的值大的话, 则 b^{x_A} 中的相对误差比 x_A 中的相对误差大得多. 例如, 若 $\text{Rel}(x_A) = 10^{-7}$, $K = 10^4$, 则 $\text{Rel}(b^{x_A}) \doteq 10^{-3}$, b^{x_A} 的精度明显低于 x_A 的精度, 这与实际上怎样计算 b^{x_A} 无关. 我们称 (2.47) 中的量为**条件数**(condition number). 它把问题输入 (此例中的 x_A) 的相对精度与输出 (即 b^{x_A}) 的相对精度关联起来. 更正式的定义, 见 Atkinson(1989, p.86). 我们将在后面继续讨论条件数的概念. ■

习题

1. 设下面给出的所有数都被正确地舍入到所示的数字位数. 对每个计算, 确定结果所在的最小区间, 注意, 这里用的是精确值而不是舍入的近似值,

(a) 1.106 2+0.947 **(b)** 23.46−12.753

(c) (2.747)(6.83) **(d)** 8.473/0.064

2. 利用区间算术运算界定 (2.45) 的误差 E, 如 (2.34) 后面和 (2.36) 后面所做的那样. 把它与 (2.45) 后面给出的结果作比较.

3. 参照 2.1 节, 利用 IEEE 算术运算标准, 求 (2.41) 中的 ε 的界.

4. 试求用 sin(1.414) 逼近 $\sin(\sqrt{2})$ 的误差界和相对误差界.

5. 在下列函数求值 $f(x_A)$ 中, 假定数 x_A 正确地舍入到所示数字位数. 利用 (2.43) 界定误差 $f(x_T) - f(x_A)$ 以及 $f(x_A)$ 的相对误差.

(a) cos(1.473) **(b)** $\tan^{-1}(2.62)$ **(c)** ln(1.471 2)

(d) $e^{2.653}$ **(e)** $\sqrt{0.042\,5}$

6. 给定函数 $f(x) = \sqrt{x}$, $x > 0$, 估计 $f(x_T) - f(x_A)$ 和 $\text{Rel}(f(x_A))$. 找出可能使精度受损的 x_T 的值 (如果存在的话).

7. 界定

$$\int_0^{\pi} \frac{t^2}{1+t^4}\mathrm{d}t - \int_0^{22/7} \frac{t^2}{1+t^4}\mathrm{d}t$$

提示: 定义

$$f(x) = \int_0^{x} \frac{t^2}{1+t^4}\mathrm{d}t$$

把 (2.43) 应用于 $f(\pi) - f(22/7)$.

8. 考虑 $f(x) = \tan x$. 设 $x_T \approx x_A$, $-\frac{1}{2}\pi < x_T, x_A < \frac{1}{2}\pi$ 且 $|x_T - x_A|$ 很小. 找出 x_A(关于 x_T) 的相对误差与 $\tan(x_A)$(关于 $\tan x_T$) 的相对误差的关系. 当 $x_T, x_A \approx \frac{1}{2}\pi$ 时, 这个关系中会发生什么情况?

9. 沿用推导 (2.37)~(2.38) 的方法, 证明

$$\text{Rel}(x_A/y_A) = \frac{\text{Rel}(x_A) - \text{Rel}(y_A)}{1 - \text{Rel}(y_A)}$$

如果 $\text{Rel}(y_A)$ 与 1 相比很小, 则

$$\text{Rel}(x_A/y_A) \approx \text{Rel}(x_A) - \text{Rel}(y_A)$$

10. 为了说明 (2.46) 和 (2.47), 比较 π^{100} 和 $\pi^{100.1}$. 尽可能精确地直接计算这些值. (大部分计算器具有足够的十进位来得到足够精确的答案). 然后利用 (2.46) 直接计算 $\text{Rel}(\pi^{100.1})$. 再给出 (2.47) 的条件数 K.

11. 设 $f(x) = (x-1)(x-2)\cdots(x-n)$. 注意 $f(1) = 0$. 对 $n = 2, 3, \cdots, 12$, 利用 (2.43) 和 $x_T = 1$ 估计 $f(1 + 10^{-4})$. 讨论这对于 $f(x)$ 求根产生的影响, 取 $n = 8$.

提示: 计算 $f'(x)$ 时不要先将 $f(x)$ 的项乘出来. 利用导数的乘积法则求 $f'(x)$, 再算出 $f'(1)$.

2.4 求　和

在许多情况下必须计算相当多的项数之总和. 我们将研究计算机求和时引进的误差以及能使最后的计算结果误差极小的方法.

用

$$S = a_1 + a_2 + \cdots + a_n = \sum_{j=1}^{n} a_j \tag{2.48}$$

表示要计算的和, 其中每个 a_j 都是浮点数. 这些数在机器中相加相当于计算一个 $n-1$ 次加法的序列, 其中每次加法可能包含一个舍入误差或截断误差. 更精确地, 定义 $a_1 + a_2$ 的浮点形式为

$$S_2 = \text{fl}(a_1 + a_2)$$

[回顾在 (2.11) 式前面的一段落介绍的 $\text{fl}(x)$ 的用法]. 接着, 定义

$$S_3 = \text{fl}(a_3 + S_2)$$

$$S_4 = \text{fl}(a_4 + S_3)$$

$$\vdots$$

$$S_n = \text{fl}(a_n + S_{n-1})$$

S_n 是 S 的计算形式.

从 (2.11) 知,

$$
\begin{aligned}
S_2 &= (a_1 + a_2)(1 + \varepsilon_2) \\
S_3 &= (a_3 + S_2)(1 + \varepsilon_3) \\
&\vdots \\
S_n &= (a_n + S_{n-1})(1 + \varepsilon_n)
\end{aligned} \tag{2.49}
$$

若假定使用 IEEE 算术运算, 每个 ε_j 满足 (2.12)~(2.16) 的界. 究竟选择哪个界与使用截断算术运算还是舍入算术运算有关, 并且也与使用单精度 IEEE 算术运算还是双精度 IEEE 算术运算有关.

将 (2.49) 中的这些关系联合起来, 并作一些处理后可得

$$
\begin{aligned}
S - S_n \approx -a_1(\varepsilon_2 + \cdots + \varepsilon_n) - a_2(\varepsilon_2 + \cdots + \varepsilon_n) - a_3(\varepsilon_3 + \cdots + \varepsilon_n) \\
-a_4(\varepsilon_4 + \cdots + \varepsilon_n) - \cdots - a_n\varepsilon_n
\end{aligned} \tag{2.50}
$$

仔细观察上式的右端可知, 为使总的误差 $S - S_n$ 达到极小, 一个自然的策略是: 求和之前先把项 $a_1, a_2, \cdots, a_n$ 按大小递增排列

$$
|a_1| \leqslant |a_2| \leqslant |a_3| \leqslant \cdots |a_n| \tag{2.51}
$$

则 (2.50) 右边具有较多个 ε_j 的项与 a_j 中较小的值相乘. 在大多数情况下这会使 $S - S_n$ 较小, 又不会增加许多附加的工作量. 这是一个合理的策略, 虽然存在更好的一些方法 (但其操作也更为复杂).

例 2.4.1 定义和 S 的项 a_j 如下: 把分数 $1/j$ 转换为十进制小数, 再把它舍入到 4 位有效数字, 并记为 a_j. 为使计算 S 中的误差更清晰, 我们使用具有 4 位浮点数有效数字的十进制机器. 从最大的项到最小的项相加 S 在表 2-9 和 2-10 中用 "LS" 表示, 而从最小的项到最大的项相加用 "SL" 表示. "真值" 列给出舍入到 4 位数字的和 S 的真值. 表 2-9 对所有求和的浮点运算使用截断的十进制机器给出求和结果. 表 2-10 对于使用舍入的机器给出类似的结果.

在两个表格中, 从最小的项到最大的项求和 S 的策略明显地优于从最大的项到最小的项求和的相反过程. 然而, 对使用舍入的机器来说, 为了看出求和次序的重要性, 就需要相当多的项数. ■

2.4.1 舍入与截断的比较

表中所示的误差在舍入和截断之间有一个比较重要的差别. 在求和计算中, 舍入的误差远远小于截断的误差. 为了解发生这种情况的原因, 返回到公式 (2.50). 作为一种典型的情况, 考虑它右边的第一项:

$$
T \equiv -a_1(\varepsilon_2 + \cdots + \varepsilon_n) \tag{2.52}
$$

假定使用例 2.4.1 的四位十进制机器的舍入. 对十进制机器, 类似于 (2.11)~(2.16) 的推导, 我们知道所有 ε_j 满足

$$-0.000\,5 \leqslant \varepsilon_j \leqslant 0.000\,5 \tag{2.53}$$

表 2-9 在使用截断的机器上计算 S

n	真值	SL	误差	LS	误差
10	2.929	2.928	0.001	2.927	0.002
25	3.816	3.813	0.003	3.806	0.010
50	4.499	4.491	0.008	4.479	0.020
100	5.187	5.170	0.017	5.142	0.045
200	5.878	5.841	0.037	5.786	0.092
500	6.793	6.692	0.101	6.569	0.224
1 000	7.486	7.284	0.202	7.069	0.417

表 2-10 在使用舍入的机器上计算 S

n	真值	SL	误差	LS	误差
10	2.929	2.929	0	2.929	0
25	3.186	3.816	0	3.817	−0.001
50	4.499	4.500	−0.001	4.498	0.001
100	5.187	5.187	0	5.187	0
200	5.878	5.878	0	5.876	0.002
500	6.793	6.794	−0.001	6.783	0.010
1 000	7.486	7.486	0	7.449	0.037

舍入误差通常可以从本质上看作是定义在上述区间里的随机变量. 因此, (2.52) 中的 ε_j 的正值和负值趋于抵消这使得它们的和 T 几乎为零. 利用概率论里的一些工具, 可以论证由 (2.52) 定义的量基本上满足下面的界限:

$$|T| \leqslant (1.49)(0.000\,5) \cdot \sqrt{n}\,|a_1|$$

即 T 值的大小大致与 $\sqrt{n}$ 成比例. 因而, (2.52) 定义的量往往较小, 除非 n 很大. 同样的结果对 (2.50) 右端总的误差也成立.

对具有截断的十进制机器, (2.53) 换成

$$-0.001 \leqslant \varepsilon_j \leqslant 0 \tag{2.54}$$

并且误差都具有同一个符号. 截断误差在这个区间中也将随机地变化. 但是现在 ε_j 的平均值将是区间的中点 $-0.000\,5$, (2.52) 的可能值是

$$-a_1(n-1)(-0.000\,5) \tag{2.55}$$

即 T 值的大小大致与 n 成比例, 而对于舍入的情况, 对应的结果与 $\sqrt{n}$ 成比例. 自然地, n 的增加远比 $\sqrt{n}$ 迅速. 因而当使用截断而不是舍入时, 误差 (2.52) 和 (2.50) 将更加迅速地变大.

例 2.4.2 我们使用可以同时进行舍入和截断的二进制计算机, 具有 6 到 7 位十进制数字的单精度. 为了说明舍入和截断之间的差别, 考虑以单精度算术运算求

$$S=\sum_{j=1}^{n}\frac{1}{j} \tag{2.56}$$

的值. 对这个计算, 误差出现在 $1/j$ 的浮点形式计算中与求和过程中. 表 2-11 给出了两种模式计算的误差. 在 "真值" 列中的值是用双精度算术运算计算 (2.56) 得到的. 所有的和式都从最小的项加到最大的项. ■

表 2-11 (2.56) 的计算: 舍入与截断的比较

n	真值	舍入误差	截断误差
10	2.928 968 25	−1.76E−7	3.01E−7
50	4.499 205 34	7.00E−7	3.56E−6
100	5.187 377 52	−4.12E−7	6.26E−6
500	6.792 823 43	−1.32E−6	3.59E−5
1 000	7.485 470 86	8.88E−8	7.35E−5

2.4.2 循环误差

累积误差的一个重要例子是循环计算 (即 Fortran 中的 Do 循环和 MATLAB 中的 for 循环) 中独立变量的计算. 假定对 $j=0,1,2,\cdots,n$, 以及给定的 $h>0$, 希望计算

$$x=a+jh \tag{2.57}$$

接着把它用于进一步的计算, 常常是计算某个 $f(x)$ 的值. 我们在此考虑的问题是, x 应该按 (2.57) 中那样计算还是在循环中利用语句

$$x=x+h \tag{2.58}$$

在循环开始前, 首先置 $x=a$. 这些计算 x 的方法在数学上是等价的, 但是它们的实际计算效果通常不等价.

计算 x 的困难一般是由 h 不能准确地存放在计算机里所引起的, 如 $h=0.1$. (2.57) 的计算涉及两个算术运算, 因此对每个 x 值只有两个截断误差或舍入误差. 与之对照, 反复使用 (2.58) 将涉及连续的加法, 事实上, (2.57) 的 x 代表它们的第 j 次加法. 当 x 的增大时, 使用 (2.58) 涉及大量的舍入误差或截断误差, 这样会导致一个不同于按 (2.57) 计算的量. 因此, (2.57) 通常是计算 x 值的优先采用的方法.

作为一个具体的例子, 我们给出以两种不同的方法计算 x 值的程序. 使用 $a=0$ 和 $h=0.1$. 为检验精度, 还利用双精度计算 x 的真值. 这个程序利用 IEEE 算术运算在一台二进制计算机上运行, 关于所选择的 $x_j=jh$ 的结果在表 2-12 中给出. (2.57) 比 (2.58) 有较大的改进.

表 2-12 求 $x=j\cdot h$ 的值, $h=0.1$

j	x	(2.57) 的误差	(2.58) 的误差
10	1	1.49E−8	−1.04E−7
20	2	2.98E−8	−2.09E−7
30	3	4.47E−8	7.60E−7
40	4	5.96E−8	1.73E−6
50	5	7.45E−8	2.46E−6
60	6	8.94E−8	3.43E−6
70	7	1.04E−7	4.40E−6
80	8	1.19E−7	5.36E−6
90	9	1.34E−7	2.04E−6
100	10	1.49E−7	−1.76E−6

2.4.3 内积的计算

下列形式之和式

$$S=a_1b_1+a_2b_2+\cdots+a_nb_n=\sum_{j=1}^{n}a_jb_j \tag{2.59}$$

称为*点积*(dot product) 或*内积*(inner product). 一般地, 项 a_j 和项 b_j 分别是向量 $\boldsymbol{A}$ 和向量 $\boldsymbol{B}$ 的分量, S 是 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的内积. 这种和式经常出现在求解某些问题中, 特别是那些涉及联立线性方程的方程组中.

若以单精度计算 S, 则每个乘法和每个加法有一个单精度舍入误差. 因此, 计算 S 中将含有 $2n-1$ 个单精度舍入误差. 这些误差的后果可用 (2.50) 的方式作分析, 并可导出计算 (2.59) 的最优策略. 作为替代, 我们考虑利用计算机双精度算术运算的更简单的方案.

用零扩展 a_j 和 b_j 的有效数字, 从而把每个 a_j 和 b_j 转换成双精度. 按双精度进行加法和乘法运算; 做完以后, 把答案舍入到单精度得到 S 的计算值. 对具有 IEEE 算术运算的机器, 在得到以单精度计算的较准确的内积时, 这个过程是一个简单而又快速的方法, 而且不必增加数组 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的存储空间. 因为不管 n 是大还是小, 都只有一个单精度舍入误差, 所以改进了计算结果的精度. 当主要的计算已经是双精度时, 需要某类扩充精度的算术运算. IEEE 浮点算术运算标准包含这样的一种扩充精度的算术运算.

MATLAB 只使用双精度 IEEE 算术运算, 故它不适合说明上面描述的想法. 下面是一个以 Fortran 编写的函数子程序 SUMPRD. 其中 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的元素利用内部构造函数 DBLE 转换成双精度, 通过附加适当数目的数字 0 把数转换成等价的双精度形式.

```
      REAL FUNCTION SUMPRD(A,B,N)
C
C     THIS CALCULATES THE INNER PRODUCT
C
C               I=N
C     SUMPRD = SUM A(I)*B(I)
C               I=1
C
C     THE PRODUCTS AND SUMS ARE DONE IN DOUBLE
C     PRECISION, AND THE FINAL RESULT IS CONVERTED
C     BACK TO SINGLE PRECISION.
C
      REAL A(*), B(*)
      DOUBLE PRECISION DSUM
C
      DSUM = 0.0D0
      DO I=1,N
        DSUM = DSUM + DBLE(A(I))*DBLE(B(I))
      END DO
      SUMPRD = DSUM
      RETURN
      END
```

习题

1. 对任意的 n, 编写一个计算机程序, 求

$$S=\sum_{j=1}^{n} a_j$$

的值, 并把它应用于下面给出的级数. 用 LS 和 SL 两种方法做计算. LS 方法由最大项到最小项进行计算与求和. SL 方法以相反的次序计算. 利用已知的答案作为计算的真值, 并把它与 LS 和 SL 得到的值作比较.

(a) $\displaystyle\sum_{j=1}^{n}\frac{1}{j(j+1)}=\frac{n}{n+1}$

(b) $\displaystyle\sum_{j=1}^{n}\frac{1}{j(j+2)}=\frac{3}{4}-\frac{2n+3}{2(n+1)(n+2)}$

(c) $\displaystyle\sum_{j=1}^{n}\frac{1}{\sqrt{j(j+1)}[\sqrt{j}+\sqrt{j+1}]}=\frac{n}{\sqrt{n+1}[\sqrt{n+1}+1]}$

使用 $n = 10, 50, 100, 500, 1\ 000, 5\ 000$.

2. 对几何级数的和

$$\sum_{j=1}^{n} x^j = \frac{1 - x^{n+1}}{1 - x}$$

重复习题 1, 取 $x = 0.01, 0.1, 0.5, 0.9, 0.99$. 讨论不同的 x 值对数值结果的影响.

3. 考虑利用部分和

$$S_n(x) = \sum_{j=0}^{n} (-1)^j \frac{x^{2j+1}}{(2j+1)!}$$

去逼近 $\sin x$. 取 $x = 0.1, 1, 10$, 用 LS 和 SL 两种方法计算 $S_n(x)$, 并把结果与 $\sin x$ 作比较. 取 $n = 10, 100, 1\ 000$.

4. 对 $n = 2, 3, 4$, 从公式 (2.49) 导出公式 (2.50).

提示: 如 (2.12)~(2.16) 指出的那样, 如果 ε_i 是小的, 则与 ε_i 相比较 $\varepsilon_i\varepsilon_j$ 非常小, 所以当 $\varepsilon_i\varepsilon_j$ 加到 ε_i 上去时可以略去不计.

5. 实施本节中给出的 Fortran 子程序 SUMPRD. 把它应用于习题 1 的求和, 把这个和看作两个数组的内积. 对 (a), 记

$$\boldsymbol{A} = \left[1, \frac{1}{2}, \frac{1}{3}, \cdots, \frac{1}{n}\right]$$

$$\boldsymbol{B} = \left[\frac{1}{2}, \frac{1}{3}, \cdots, \frac{1}{n+1}\right]$$

还有, 只利用单精度算术运算计算同样的内积并把它与利用 SUMPRD 得到的答案作比较.

6. 对 $a_i = \mathrm{fl}(a_i)$, $i = 1, 2, \cdots, n$, 考虑求 $p = a_1 a_2 \cdots a_n$ 的值. 定义

$$p_2 = \mathrm{fl}(a_1 a_2), \quad p_3 = \mathrm{fl}(p_2 a_3), \quad \cdots, \quad p_n = \mathrm{fl}(p_{n-1} a_n)$$

用类似于导出 (2.50) 所用的方法, 导出 $p_n - p$ 的估计和 $\mathrm{Rel}(p_n)$ 的估计, 指出在构成 $p_2, \cdots, p_n$ 中出现的舍入误差或截断误差的影响.

第3章 求 根

计算方程

$$f(x) = 0 \tag{3.1}$$

的根是应用数学中常见的问题. 我们将探讨求解这种方程的几个简单的数值方法, 同时也考虑一些可能会碰到的困难. 我们从金融规划中的一个简单例子开始讨论.

例 假设提前为退休做打算, 我们可能会选择投资, 希望退休以后能有一个满意的收入. 这个过程的简化形式便产生了*养老保险年金*(annuity) 的概念, 我们在此简略地讨论它.

假设在 N_{in} 段时间的每段一开始, 把现金 P_{in} 存入一个储蓄账户. 例如, 如果按月存入, 则用退休前的年数乘以 12 就得到 N_{in}. 经过 N_{in} 次存款之后, 假设在以后的 N_{out} 段时间的每段, 从账户中支取的金额为 P_{out}. 在第 $N_{\text{in}}+1$ 段时间开始支取第一笔资金. 在 $N_{\text{in}}+N_{\text{out}}$ 段时间内, 账户中要以每期为 r 的复利率增加利息. 例如, 如果账户中的利息是由年利率为 6% 的月复利率产生, 则 r=0.06/12=0.005. 假设支取最后一笔资金后账户余额为零, 试问 N_{in}, N_{out}, P_{in}, P_{out} 和 r 之间的关系是什么?

在 N_{in} 段时间结束时, 账户中的金额是

$$S_{\text{in}} \equiv P_{\text{in}}\{(1+r)^{N_{\text{in}}} + \cdots + (1+r)\} = P_{\text{in}}(1+r)\frac{(1+r)^{N_{\text{in}}}-1}{r}$$

在第 $N_{\text{in}}+1$ 段时间开始, 支取 N_{out} 笔款项中的第一笔, 每笔金额是 P_{out}; 在第 $N_{\text{in}}+N_{\text{out}}$ 段时间开始, 支取最后一笔款项, 此时账户余额变为零. 支取第一笔款项后账户余额是 $S_{\text{in}}-P_{\text{out}}$; 在 N_{out} 次支取之后, 账户余额 (包括这期间产生的复利总额) 为

$$(1+r)^{N_{\text{out}}-1}S_{\text{in}} - P_{\text{out}}\frac{(1-r)^{N_{\text{out}}}-1}{r} = 0$$

结合 S_{in} 的公式并简化之, 得到方程

$$f(r) \equiv P_{\text{in}}[(1+r)^{N_{\text{in}}}-1] - P_{\text{out}}[1-(1+r)^{-N_{\text{out}}}] = 0 \tag{3.2}$$

给定 P_{in}, P_{out}, N_{in}, N_{out} 和 r 中的任意四项, 就可以求第五项. 这些问题可以直接计算, 但下列情况除外: 计算 r 是一个求根问题, 这需要使用数值方法. 我们将在后面的习题中讨论这个计算 r 的求根问题. ■

方程 (3.1) 中的函数 $f(x)$ 一般至少有一阶连续的导数, 而且我们对所求方程的根常有某些估计. 利用这些信息, 求解 (3.1) 的大部分数值方法计算一系列越来越准确的根的近似值. 这些方法称为**迭代法**(iteration method). 本章前三节讨论三种不同的求根方法, 在 3.4 节中给出单点迭代法的一般理论. 对某些特殊函数 $f(x)$ 进行方程求根时所遇到的困难将在 3.5 节中讨论.

3.1 对 分 法

本章假设 $f(x)$ 是实值函数且 x 是实变量. 设 $f(x)$ 是区间 $a \leqslant x \leqslant b$ 上的连续函数且

$$f(a)f(b) < 0 \tag{3.3}$$

则 $f(x)$ 在 $[a, b]$ 上变号, 所以 $f(x) = 0$ 在这个区间上至少有一个根. 最简单的数值求根方法是反复对分区间 $[a, b]$, 保留 $f(x)$ 改变符号的那一半区间. 这个过程称为**对分法**(bisection method). 对分法确保收敛于一个根, 此根记为 α.

更精确地说, 假设区间 $[a, b]$ 满足 (3.3), 并且给定误差容限 $\varepsilon > 0$, 则对分法由下列步骤组成.

B1. 定义 $c = (a + b)/2$.

B2. 如果 $b - c \leqslant \varepsilon$, 则认可 c 为根, 终止计算.

B3. 如果 $\text{sign}[f(b)] \cdot \text{sign}[f(c)] \leqslant 0$, 则取 $a = c$. 否则, 取 $b = c$, 返回 B1.

每一次从步骤 B1 到 B3 的循环都把区间 $[a, b]$ 分为两半. 最终 B2 成立并且满足条件 $|\alpha - c| \leqslant \varepsilon$, 这一点的正确性可以从后文中给出的 (3.7) 中看出. 为了避免 $f(b)$ 和 $f(c)$ 相乘可能产生的上溢和下溢, 在步骤 B3 中我们检验 $\text{sign}[f(b)] \cdot \text{sign}[f(c)]$ 的符号而不是检验 $f(b)f(c)$ 的符号.

例 3.1.1 求方程

$$f(x) \equiv x^6 - x - 1 = 0 \tag{3.4}$$

的最大根, 精度在 $\varepsilon = 0.001$ 之内. 利用函数 $f(x)$ 的图形, 易知 $1 < \alpha < 2$. 选取 a=1, b=2, 有 $f(a) = -1, f(b) = 61$, (3.3) 式满足. 算法 B1 到 B3 的计算结果如表 3-1 所示. 表中的 n 表示相关的行对应于从步骤 B1 到 B3 的迭代次数. ■

误差界

设 a_n, b_n 和 c_n 分别表示 a, b 和 c 的第 n 次的计算值. 容易得到

$$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n), \quad n \geqslant 1 \tag{3.5}$$

由此直接可得

$$b_n - a_n = \frac{1}{2^{n-1}}(b - a), \quad n \geqslant 1 \tag{3.6}$$

表 3-1 方程 (3.4) 的对分法

n	a	b	c	$b-c$	$f(c)$
1	1.000 0	2.000 0	1.500 0	0.500 0	8.890 6
2	1.000 0	1.500 0	1.250 0	0.250 0	1.564 7
3	1.000 0	1.250 0	1.125 0	0.125 0	−0.097 7
4	1.125 0	1.250 0	1.187 5	0.062 5	0.616 7
5	1.125 0	1.187 5	1.156 2	0.031 2	0.233 3
6	1.125 0	1.156 2	1.140 6	0.015 6	0.061 6
7	1.125 0	1.140 6	1.132 8	0.007 8	−0.019 6
8	1.132 8	1.140 6	1.136 7	0.003 9	0.020 6
9	1.132 8	1.136 7	1.134 8	0.002 0	0.000 4
10	1.132 8	1.134 8	1.133 8	0.000 98	−0.009 6

其中 $b-a$ 表示原始区间的长度. 由于根 α 在区间 $[a_n, c_n]$ 或区间 $[c_n, b_n]$ 中, 所以

$$|\alpha - c_n| \leqslant c_n - a_n = b_n - c_n = \frac{1}{2}(b_n - a_n) \tag{3.7}$$

这是前面算法的步骤 B2 中 c_n 的误差界. 把它与 (3.6) 式结合起来, 得到

$$|\alpha - c_n| \leqslant \frac{1}{2^n}(b-a) \tag{3.8}$$

这表明当 $n \to \infty$ 时, 迭代 c_n 收敛于 α.

为了知道所需迭代的次数, 假设我们希望达到下面的精度:

$$|\alpha - c_n| \leqslant \varepsilon$$

由 (3.8) 知, 当

$$\frac{1}{2^n}(b-a) \leqslant \varepsilon$$

时上式成立. 两边取对数, 可解得

$$n \geqslant \frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln 2} \tag{3.9}$$

在解方程 (3.4) 的例子中, 相应的结果是

$$n \geqslant \frac{\ln\left(\frac{1}{0.001}\right)}{\ln 2} \doteq 9.97$$

因此, 取整数后知迭代次数 $n = 10$.

对分法有好几个优点. 主要的优点是确保收敛性. 另外, (3.7) 给出的误差界, 每次迭代都确保减少一半. 其他许多数值方法具有不同的误差下降速率, 对某些方程来说, 这些方法可能比对分法差. 对分法的主要缺点是它通常比其他大部分方法

收敛得慢. 对于具有连续导数的函数 $f(x)$ 来说, 其他方法一般收敛得更快. 这些方法可能不会总是收敛的, 但是当它们收敛时, 其收敛速度几乎总是比对分法快很多.

MATLAB 程序: 对分法的实施 下面给出一个实现对分法的bisect程序. 关于程序的目的和用途, 在程序开头有明确的说明. 程序的形式相当标准. 函数 f 作为内部函数(internal function) 给出, 即存放在这个文件之外的 MATLAB 程序不能识别这个函数. 程序包含对分法中间步骤的一个内部打印, 这一步可以省略. 程序结束时, 可以打印出根的近似值及其误差界.

就计算机算术运算的有效数字长度而言, 这个程序没有检验给定的误差界ep是否实用. 这样做是为了使程序简单. 不过, 一个计算机程序库中实际的求根程序应该包含这样的检验. 在 MATLAB 中, 可以用给定的机器常数eps来构造这样的一个检验.

```
function [root,error_bound] = bisect(a0,b0,ep,max_iterate)
%
% function bisect(a0,b0,ep,max_iterate)
%
% This is the bisection method for solving an equation f(x)=0.
%
% The function f is defined below by the user.  The function f is
% to be continuous on the interval [a0,b0], and it is to be of
% opposite signs at a0 and b0.  The quantity ep is the error
% tolerance.  The parameter max_iterate is an upper limit on the
% number of iterates to be computed.
%
% This program guarantees ep as an error bound for the computed
% root provided:  (1) the restrictions on the given function f
% and the initial [a0,b0] are satisfied; (2) ep is not too small
% when the machine epsilon is taken into account; and (3) the
% number of iterates computed is at most max_iterate.  Only
% some of these conditions are checked in the program!
%
% For the given function f(x), an example of a calling sequence
% might be the following:
%       [root,error_bound] = bisect(1,1.5,1.0E-6,10)
%
% The following is printed for each iteration the values of
%       count, a, b, c, f(c), (b-a)/2
% with c the current iterate and (b-a)/2 the error bound for c.
% The variable count is the index of the current iterate.  Tap
% the carriage return to continue with the iteration.
f a0 >= b0
 disp('a0 < b0 is not true.  Stop!')
```

```
 return
end

format short e
a = a0; b = b0;
fa = f(a); fb = f(b);

if sign(fa)*sign(fb) > 0
  disp('f(a0) and f(b0) are of the same sign.  Stop!')
  return
end

c = (a+b)/2;
it_count = 0;
fprintf('\n  it_count  a  b  c  f(c)  b-c\n')
while b-c > ep & it_count < max_iterate
  it_count = it_count + 1;
  fc = f(c);
% Internal print of bisection method.  Tap the carriage
% return key to continue the computation.
  iteration = [it_count a b c fc b-c]
  if sign(fb)*sign(fc) <= 0
    a = c;
    fa = fc;
  else
    b = c;
    fb = fc;
  end
  c = (a+b)/2;
  pause
end

format long
root = c
format short e
error_bound = b-c

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = f(x)
%
% function to define equation for rootfinding problem.

value = x.^6 - x - 1;
```

习题

1. 用对分法求下列方程所示的根. 误差界为 $\varepsilon = 0.000\,1$.

(a) $x^3-x^2-x-1=0$ 的实根.

(b) $x=1+0.3\cos(x)$ 的根.

(c) $\cos(x)=\frac{1}{2}+\sin(x)$ 的最小的正根.

(d) $x=\mathrm{e}^{-x}$ 的根.

(e) $\mathrm{e}^{-x}=\sin(x)$ 的最小的正根.

(f) $x^3-2x-2=0$ 的实根.

(g) $x^4-x-1=0$ 的所有实根.

2. 为了帮助确定 $x=\tan(x)$ 的根, 画出 $y=x$ 和 $y=\tan(x)$ 的图形, 查看两条曲线的交点.

(a) 求出 $x=\tan(x)$ 的最小的非零正根, 精度为 $\varepsilon=0.000\ 1$.

注意: 所求的根大于$\frac{\pi}{2}$.

(b) 求出 $x=\tan(x)$ 距离 x=100 最近的根.

3. 在方程 (3.2) 中, 设 $P_{\text{in}}=1\ 000$, $P_{\text{out}}=20\ 000$, $N_{\text{in}}=30$, $N_{\text{out}}=20$, 以 $\varepsilon=0.000\ 1$ 的精度求 r.

4. 证明: 对于任意实常数 c 和 d, 方程 $x=c+d\cos(x)$ 至少有一个根.

提示: 找一个区间 $[a, b]$, 使得 $f(x)=x-c-d\cos(x)$ 在该区间上变号.

5. 利用对分法, 实施`bisect`程序或者编写自己设计的程序. 用你所选程序求解习题 1 和习题 2 中的方程, 精度取 $\varepsilon=10^{-5}$.

6. 利用对分法和 $f(x)$ 的图形, 求方程 $f(x)=32x^6-48x^4+18x^2-1$ 的全部根. 根的精确值是 $\cos\left[(2j-1)\frac{\pi}{12}\right], j=1,2,\cdots,6$.

7. 用习题 5 的程序, 求解方程

$$f(x)\equiv x^3-3x^2+3x-1=0$$

精度取 $\varepsilon=10^{-6}$. 试对计算 $f(x)$ 值的不同方法进行实验, 例如, 使用 (i) 方程所给形式, (ii) 方程所有项顺序颠倒的形式, (iii) 方程的嵌套形式

$$f(x)=-1+x(3+x(-3+x))$$

尝试不同的初始区间 $[a, b]$, 例如 [0, 1.5], [1.5, 2.0] 和 [0.5, 1.1]. 解释你的结果. 注意, $\alpha=1$ 是 $f(x)=0$ 的唯一的根.

8. 多项式

$$f(x)=x^4-5.4x^3+10.56x^2-8.954x+2.795\ 1$$

在区间 [1, 1.2] 中有根 α, 重复习题 7, 但是区间 $[a, b]$ 的选择与根 α 的位置有关.

9. 设对分法中初始区间的长度为 $b-a$=3. 为了得到误差容限为 $\varepsilon=10^{-9}$ 的近似根, 试求对分法必须计算的中点 c_n 的个数.

10. 考虑方程 $\mathrm{e}^{-x}=\sin x$. 找一个含方程最小正根的区间 $[a, b]$. 为了得到准确到误差容限为 10^{-10} 的近似根, 估计所需要的中点 c 的个数.

11. 设 α 是方程

$$f(x) \equiv 1 - x + \sin x = 0$$

的最小正根. 找一个含有 α 的区间 $[a, b]$, 且对分法用于该区间上时收敛于 α. 在 5×10^{-8} 的精度内, 估计求 α 所需的迭代次数.

12. 设 α 是方程

$$x = \frac{3}{1 + x^4}$$

的唯一根. 找一个含有 α 的区间 $[a, b]$, 且对分法在该区间上收敛于 α. 然后在 5×10^{-8} 的精度内, 估计求 α 所需的迭代次数.

13. 设 α 是方程 $f(x) \equiv \mathrm{e}^x - x - 2 = 0$ 的最大根. 找一个含有 α 的区间 $[a, b]$, 且对分法用于该区间上时收敛于 α. 然后在 5×10^{-8} 的精度内, 估计求 α 所需的迭代次数.

14. 想象求一个满足 $1 < \alpha < 2$ 的根 α, 并且使用有效数字为 m 位二进数的二进制计算机. 试问求 α 的近似解时合理的最小误差容限是多少? 如果最初的区间是 $[a, b]=[1, 2]$, 试问求 α 的具有计算机最大可能精度的近似解 c_n 需要对分区间多少次?

15. 设 $f(x) = 1 - zx$, 其中 $z > 0$. 求解 $f(x) = 0$ 等价于计算 $1/z$, 即做除法运算.

(a) 试给出一个区间 $[a, b]$, 或计算此区间的方法, 保证它包含 $1/z$. 在定义或计算 $[a, b]$ 时不能使用除法.

(b) 假设 $1 < z < 2$. 利用某个包含 $1/z$ 的区间, 为了得到精度在 2^{-25} 之内的估计, 给出所需细分区间的次数 n.

(c) 对一般的 $z > 0$, 考虑按 IEEE 标准浮点数算术运算的单精度算术运算计算 $1/z$. 应用对分法, 给出计算 $1/z$ 的方法使其达到算术运算的最高精度.

3.2 牛 顿 法

考虑图 3-1 所示 $y = f(x)$ 的样本图形. 根 α 对应于该图形与 x 轴的交点. 我们往往会有根 α 的一个估计, 把它记为 x_0. 为了改进这个估计, 考虑经过点 $(x_0, f(x_0))$ 的图形的切线. 如果 x_0 靠近 α, 则对于 α 附近的点 x 来说, 该切线几乎与 $y = f(x)$ 的图形重合. 因此切线的根应该几乎等于 α, 这里把这个根记为 x_1.

为了找到 x_1 的公式, 考察 $y = f(x)$ 的图形在点 $(x_0, f(x_0))$ 的切线方程. 它就是 $y = p_1(x)$ 的图形, 其中 $p_1(x)$ 为线性泰勒多项式

$$p_1(x) = f(x_0) + f'(x_0)(x - x_0)$$

由定义知, x_1 是 $p_1(x)$ 的根. 求解

$$f(x_0) + f'(x_0)(x_1 - x_0) = 0$$

得到

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{3.10}$$

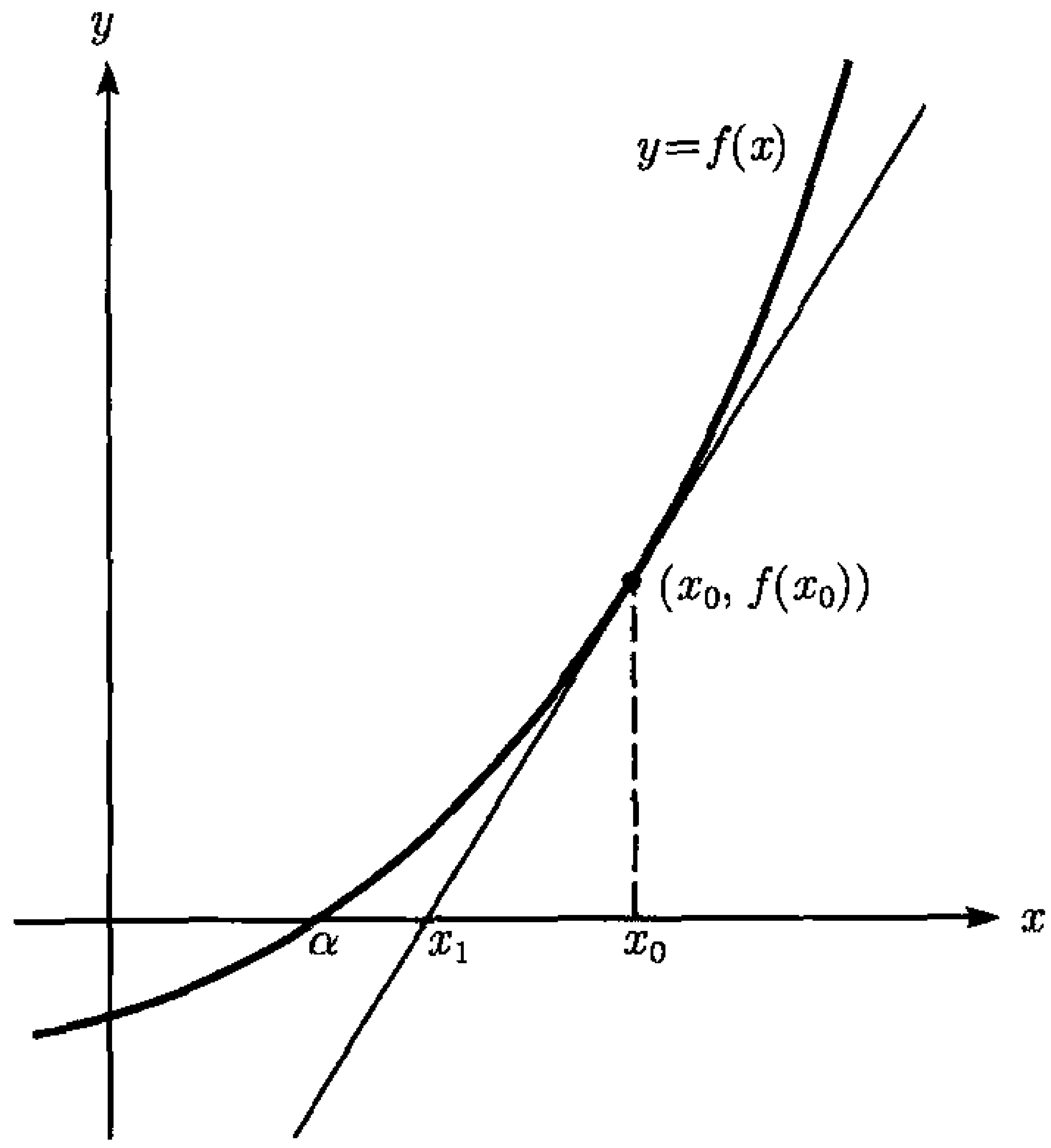


图 3-1 牛顿法图解

我们期望 x_1 比 x_0 更接近 α. 把 x_1 作为初始估计, 可以重复上述过程而得到新的估计

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

不断重复这个过程, 得到一连串的数 $x_1, x_2, x_3, \cdots$, 希望它们能趋向于根 α. 称这些数为迭代值, 它们由下列一般的**迭代公式**(iteration formula) 递归地定义

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \cdots \tag{3.11}$$

这是解方程 $f(x) = 0$ 的**牛顿法**(Newton method), 也称为**牛顿–拉夫森方法**(Newton-Raphson method).

例 3.2.1 用牛顿法求解方程 (3.4), 例 3.1.1 中曾用此方程说明对分法. 这里

$$f(x) = x^6 - x - 1, \quad f'(x) = 6x^5 - 1$$

由下式给出迭代

$$x_{n+1} = x_n - \frac{x_n^6 - x_n - 1}{6x_n^5 - 1}, \quad n \geqslant 0 \tag{3.12}$$

取初始猜测 $x_0 = 1.5$. 计算结果如表 3-2 所示. "$x_n - x_{n-1}$" 所在的列是误差 $\alpha - x_{n-1}$ 的估计; 本节后面给出它的证明.

根的精确值是 $\alpha \doteq 1.134\ 724\ 138$, x_6 与 α 有 9 位有效数字相同. 请你把牛顿法的计算结果与表 3-1 所示的对分法计算结果进行比较. 注意到牛顿法在最初阶段收敛较慢, 而当迭代不断靠近根时, 收敛速度也在增加. ■

表 3-2 解 $x^6 - x - 1 = 0$ 的牛顿法

n	x_n	$f(x_n)$	$x_n - x_{n-1}$	$\alpha - x_{n-1}$
0	1.5	8.89E+1		
1	1.300 490 88	2.54E+1	−2.00E−1	−3.65E−1
2	1.181 480 42	5.38E−1	−1.19E−1	−1.66E−1
3	1.139 455 59	4.92E−2	−4.20E−2	−4.68E−2
4	1.134 777 63	5.50E−4	−4.68E−3	−4.73E−3
5	1.134 724 15	7.11E−8	−5.35E−5	−5.35E−5
6	1.134 724 14	1.55E−15	−6.91E−9	−6.91E−9

例 3.2.2 现在考虑在一些早期的计算机上实现除法运算的过程. 这些计算机具有加法、减法和乘法运算的硬件, 但除法只能靠软件实现. 某些当代的超级计算机也用下面出现的迭代 (3.14).

假设我们要计算 a/b, 这可通过 a 和 $1/b$ 相乘来做到, 而计算 $1/b$ 可利用牛顿法. 为此我们求解方程

$$f(x) \equiv b - \frac{1}{x} = 0 \tag{3.13}$$

其中 $b > 0$. 方程的根是 $\alpha = 1/b$. 因为

$$f'(x) = \frac{1}{x^2}$$

所以解方程 (3.13) 的牛顿迭代公式为

$$x_{n+1} = x_n - \frac{b - \frac{1}{x_n}}{\frac{1}{x_n^2}}$$

化简得到

$$x_{n+1} = x_n(2 - bx_n), \quad n \geqslant 0 \tag{3.14}$$

上述过程只涉及乘法和减法, 在早期计算机上可以毫无困难地实现. 当然, 初始猜测应选择 $x_0 > 0$. 关于误差, 可以证明

$$\text{Rel}(x_{n+1}) = [\text{Rel}(x_n)]^2, \quad n \geqslant 0 \tag{3.15}$$

其中

$$\text{Rel}(x_n) = \frac{\alpha - x_n}{\alpha}$$

是 x_n 关于 $\alpha = 1/b$ 的相对误差. 从 (3.15) 知, 我们需要

$$|\text{Rel}(x_0)| < 1$$

否则, 当 n 增大时 x_n 的误差将不会递减至零. 这个条件意味着

$$-1 < \frac{\frac{1}{b} - x_0}{\frac{1}{b}} < 1$$

它等价于条件

$$0 < x_0 < \frac{2}{b} \tag{3.16}$$

迭代 (3.14) 收敛于 $\alpha = 1/b$ 当且仅当初始猜测 x_0 满足 (3.16). 图 3-2 展示了一个实例. 从图 3-2 可以看出条件 (3.16) 的重要性, 因为如果这一条件不满足, 则 x_1 与所有的后继迭代项都是负的.

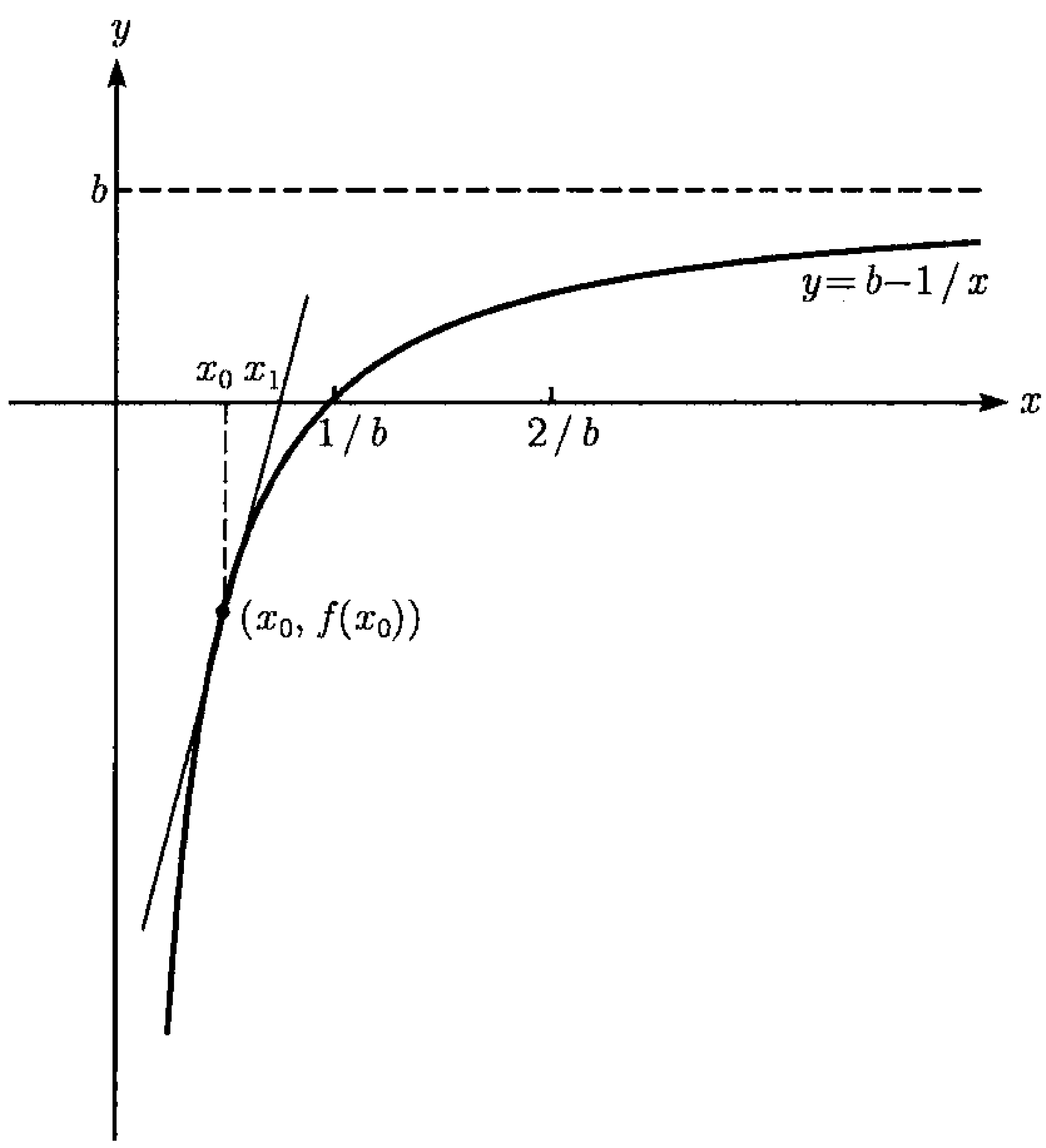


图 3-2 $b - 1/x = 0$ 的迭代解

一旦我们有一个较精确的初始估计, (3.15) 的结果表明收敛速度非常快. 例如, 假设 $|\text{Rel}(x_0)| = 0.1$, 即 x_0 的误差为 10%. 由 (3.15) 知

$$\text{Rel}(x_1) = 10^{-2}, \quad \text{Rel}(x_2) = 10^{-4},$$
$$\text{Rel}(x_3) = 10^{-8}, \quad \text{Rel}(x_4) = 10^{-16} \tag{3.17}$$

对于许多问题来说 x_3 和 x_4 已经足够精确了. ■

3.2.1 误差分析

假设 $f(x)$ 对于根 α 周围某个区间中的所有 x 至少具有二阶连续的导数. 再假设

$$f'(\alpha) \neq 0 \tag{3.18}$$

也就是说当 $y = f(x)$ 的图形与 x 轴在 $x = \alpha$ 点相交时, 图形与 x 轴是不相切的. $f'(\alpha) = 0$ 的情形将在 3.5 节中探讨. 另外注意到, 把 (3.18) 和 $f'(x)$ 的连续性结合起来可推出, 对 α 附近的所有 x 都有 $f'(x) \neq 0$. 利用泰勒定理可得

$$f(\alpha) = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{1}{2}(\alpha - x_n)^2 f''(c_n)$$

其中 c_n 是 α 与 x_n 之间的一个未知点. 由假设知 $f(\alpha) = 0$, 等式两端除以 $f'(x_n)$ 得到

$$0 = \frac{f(x_n)}{f'(x_n)} + \alpha - x_n + (\alpha - x_n)^2 \frac{f''(c_n)}{2f'(x_n)}$$

由 (3.11) 知, 右端第一项是 $x_n - x_{n+1}$, 于是

$$0 = x_n - x_{n+1} + \alpha - x_n + (\alpha - x_n)^2 \frac{f''(c_n)}{2f'(x_n)}$$

解出 $\alpha - x_{n+1}$, 得

$$\alpha - x_{n+1} = (\alpha - x_n)^2 \left[\frac{-f''(c_n)}{2f'(x_n)}\right] \tag{3.19}$$

此公式表明 x_{n+1} 的误差几乎与 x_n 的误差的平方成比例. 这说明如果初始误差足够小, 则后续迭代中的误差会迅速下降, 正如 (3.17) 所示. 公式 (3.19) 还可以用来给出牛顿法收敛的一个严格的数学证明, 但我们把它略去.

例 3.2.3　考虑先前的迭代 (3.12), $f''(x) = 30x^4$. 若 x_n 在根 α 邻近, 则有

$$\frac{-f''(c_n)}{2f'(x_n)} \approx \frac{-f''(\alpha)}{2f'(\alpha)} = \frac{-30\alpha^4}{2(6\alpha^5 - 1)} \doteq -2.42$$

因此关于 (3.12) 中的误差, 有

$$\alpha - x_{n+1} \approx -2.42(\alpha - x_n)^2 \tag{3.20}$$

此结果说明了表 3-2 中最终迭代的快速收敛性. 例如, 考虑 $n = 3$ 的情形, $\alpha - x_3 \doteq -4.73\text{E} - 3$. 于是 (3.20) 给出

$$\alpha - x_4 \doteq -2.42(4.73\text{E} - 3)^2 \doteq -5.42\text{E} - 5$$

这与实际误差 $\alpha - x_4 \doteq -5.35\text{E} - 5$ 很接近. ■

如果迭代值 x_n 在根 α 附近, 则 (3.19) 右端的乘数可写为

$$\frac{-f''(c_n)}{2f'(x_n)} \approx \frac{-f''(\alpha)}{2f'(\alpha)} \equiv M \tag{3.21}$$

因此

$$\alpha - x_{n+1} \approx M(\alpha - x_n)^2, \quad n \geqslant 0 \tag{3.22}$$

两端乘以 M 得到

$$M(\alpha - x_{n+1}) \approx [M(\alpha - x_n)]^2 \tag{3.23}$$

假设所有迭代都在 α 附近, 则我们可归纳地证明

$$M(\alpha - x_n) \approx [M(\alpha - x_0)]^{2^n}, \quad n \geqslant 0$$

由于我们要求 $\alpha - x_n$ 收敛于零, 故必须假定

$$|M(\alpha - x_0)| < 1$$

$$|\alpha - x_0| < \frac{1}{|M|} = \left|\frac{2f'(\alpha)}{f''(\alpha)}\right| \tag{3.24}$$

如果 $|M|$ 很大, 则为了保证收敛, x_0 必须选得很靠近 α. 此时, 对分法或许是一种更容易使用的方法. 习题 5 就是这种情况的例子.

确定牛顿法是否收敛, x_0 的选取是非常重要的. 遗憾的是, 并不存在单个总是有效的策略来选取 x_0. 大多数情况下, x_0 的选择可借助于产生求根问题的物理背景. 在另一些情况, 最初几步迭代可能要结合对分法, 很可能需要画出 $y = f(x)$ 的图形.

3.2.2 误差估计

计算迭代序列 x_n 时, 我们希望能估计它们的精度以便知道何时停止迭代. 由于 $f(\alpha) = 0$, 为了估计 $\alpha - x_n$, 根据中值定理, 有

$$f(x_n) = f(x_n) - f(\alpha) = f'(\xi_n)(x_n - \alpha)$$

其中 ξ_n 位于 x_n 和 α 之间. 假如 x_n 非常接近 α 使得 $f'(x_n) \doteq f'(\xi_n)$, 则可得到

$$\alpha - x_n = \frac{-f(x_n)}{f'(\xi_n)} \approx \frac{-f(x_n)}{f'(x_n)}$$

根据 (3.11), 上式可改写为

$$\alpha - x_n \approx x_{n+1} - x_n \tag{3.25}$$

这是牛顿法的标准误差估计公式, 它通常是比较精确的. 但是, 当 $f'(\alpha) = 0$ 时该公式无效, 这种情况在本章 3.5 节中讨论.

例 3.2.4 考察表 3-2 中元素 x_3 的误差

$$\alpha - x_3 \doteq -4.73\mathrm{E}-3, \quad x_4 - x_3 \doteq -4.68\mathrm{E}-3 \tag{3.26}$$

此结果显示出公式 (3.25) 的精确性. ■

MATLAB 程序: 牛顿法的实施 `Newton`程序用于实现牛顿法, 它包括误差估计和预防无限循环的措施. 输入参数`max_iterate`是所计算的迭代次数上界, 这个参数可以防止发生无限循环现象.

```
function root = newton(x0,error_bd,max_iterate)
%
% function newton(x0,error_bd,max_iterate)
%
% This is Newton's method for solving an equation f(x) = 0.
%
% The functions f(x) and deriv_f(x) are given below.
% The parameter error_bd is used in the error test for the
% accuracy of each iterate.  The parameter max_iterate
% is an upper limit on the number of iterates to be
% computed.  An initial guess x0 must also be given.
%
% For the given function f(x), an example of a calling sequence
% might be the following:
%       root = newton(1,1.0E-12,10)
%
% The program prints the iteration values
%       iterate_number, x, f(x), deriv_f(x), error
% The value of x is the most current initial guess, called
% previous_iterate here, and it is updated with each iteration.
% The value of error is
%       error = newly_computed_iterate - previous_iterate
% and it is an estimated error for previous_iterate.
% Tap the carriage return to continue with the iteration.
format short e
error = 1;
it_count = 0;
fprintf('\n  it_count  x  f(x)  df(x)  error  \n')

while abs(error) > error_bd & it_count < max_iterate
  fx = f(x0);
  dfx = deriv_f(x0);
  if dfx == 0
```

```
    disp('The derivative is zero.  Stop')
    return
  end
  x1 = x0 - fx/dfx;
  error = x1 - x0;
% Internal print of the Newton method.  Tap the carriage
% return key to continue the computation.
  iteration = [it_count x0 fx dfx error]
  pause
  x0 = x1;
  it_count = it_count + 1;
end

if it_count >= max_iterate
  disp('The number of iterates calculated exceeded')
  disp('max_iterate.  An accurate root was not')
  disp('calculated.')
else
  format long
  root = x1
  format short
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = f(x)
%
% function to define equation for rootfinding problem.
%
value = x.^6 - x - 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = deriv_f(x)
%
% Derivative of function defining equation for
% rootfinding problem.
%
value = 6*x.^5 - 1;
```

习题

1. 用两个初始猜测 $x_0 = 1.0$ 和 $x_0 = 2.0$ 执行牛顿迭代 (3.12). 将计算结果与表 3-2 中的结果进行比较.
2. 用牛顿法求 3.1 节习题 1 中的方程的根, 用 $\varepsilon = 10^{-6}$ 作为误差容限.
3. (a) 在大多数计算机中, $\sqrt{a}$ 的计算建立在牛顿法的基础上. 给出求解 $x^2 - a = 0$ 的牛顿

迭代公式, 证明它可以写为下列形式

$$x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right),\quad n\geqslant 0$$

(b) 导出误差公式和相对误差公式

$$\sqrt{a}-x_{n+1}=-\frac{1}{2x_n}(\sqrt{a}-x_n)^2$$

$$\text{Rel}(x_{n+1})\ =-\frac{\sqrt{a}}{2x_n}[\text{Rel}(x_n)]^2$$

提示: 应用 (3.19) 式.

(c) 对 $\sqrt{a}$ 附近的 x_0, 上述最后一个公式变为

$$\text{Rel}(x_{n+1})\approx-\frac{1}{2}\left[\text{Rel}(x_n)\right]^2,\quad n\geqslant 0$$

假设 $\text{Rel}(x_0)=0.1$, 利用上述公式估计 x_1, x_2, x_3 和 x_4 的相对误差.

4. 给出求解 $\sqrt[m]{a}$ 的牛顿法, 其中 $a>0$ 且 m 是一个正整数. 应用它求解 $\sqrt[m]{2}$, 其中 $m=3,4,5,6,7,8$, 保留 6 位有效数字.

提示: 求解 $x^n-a=0$.

5. (a) 重复 3.1 节的习题 2, 求出 $x=\tan(x)$ 的根. 用 $\varepsilon=10^{-6}$ 作为误差容限.

(b) 用牛顿法很难求出位于 100 附近的根. 为解释其原因, 计算 (3.21) 式中的 M, 并把它用于关于 x_0 的条件 (3.24).

6. 方程

$$f(x)\equiv x+\mathrm{e}^{-Bx^2}\cos(x)=0,\quad B>0$$

有唯一的根, 它位于区间 $(-1,0)$ 中. 利用牛顿法尽可能精确地求其根. 分别取 $B=1,5,10,25,50$. 选取 $x_0=0$ 和其他一些值作为初值, 解释 B 取较大值时在迭代中观察到的情形.

提示: 画出 $f(x)$ 的图形以更好地了解函数的变化情况.

7. 如 (3.26) 式中所做的那样, 检查表 3-2 中所有元素的误差逼近 (3.25) 的精度.

8. 利用迭代 (3.14) 计算 1/3. 取 $x_0=0.2$. 给出 x_1, x_2, x_3 和 x_4 的误差.

9. 推导公式 (3.15), 并用它证明 $\text{Rel}(x_n)=[\text{Rel}(x_0)]^{2^n}$.

提示: 利用

$$\text{Rel}(x_{n+1})=\frac{\alpha-x_{n+1}}{\alpha}=1-bx_{n+1}$$

用 (3.14) 替代 x_{n+1}, 然后把计算结果与 $[\text{Rel}(x_n)]^2$ 作比较.

10. 用牛顿法在计算机上解方程

$$x^3-3x^2+3x-1=0$$

回顾 3.1 节习题 7, 选取不同的初值 x_0 进行试验, 同时对计算 $f(x)$ 和 $f'(x)$ 值的不同方法进行试验. 注意迭代中出现的任何异常情形.

11. 对下列方程重复习题 10:

$$x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.795\,1 = 0$$

寻找位于 [1, 1.2] 中的根 α.

12. 回顾 1.3 节中多项式嵌套求值的内容, 特别是 (1.35)~(1.38) 以及 1.3 节中的习题 8. 利用这些材料, 编写一个求多项式 $p(x)$ 根的牛顿程序, 用这些先前的材料有效地计算 $p(x)$ 和 $p'(x)$ 的值. 把程序应用于下列多项式方程, 求出其最大的正根:

(a) $x^3 - x^2 - x - 1 = 0$ **(b)** $32x^6 - 48x^4 + 18x^2 - 1 = 0$

利用 (1.37) 和 (1.38) 中出现的多项式 $q(x)$, 我们知道根 α 满足 $p(x) = (x-\alpha)q(x)$. 因此, 可用 $q(x)$ 得到 $p(x) = 0$ 的另外的根. 这个方法称为**多项式压缩**(polynomial deflation).

13. 考虑应用牛顿法求 $\sin(x) = 0$ 的根 $\alpha = 0$. 找到一个区间 $[-r, r]$ 使得对于 $[-r, r]$ 中任意的 x_0, 牛顿迭代法在该区间上都收敛于 α. 要找出尽可能大的 r.

提示: 画出 $y = \sin(x)$ 的图形, 并在图形上描出 x_0 和 x_1 的位置.

3.3 割 线 法

牛顿法基于用切线去逼近 $y = f(x)$ 的图形, 然后用该直线的根作为 $f(x)$ 的根 α 的近似. 从这个视角看, 其他逼近 $y = f(x)$ 的直线也可导致逼近 $f(x)$ 的根的方法. 其中一种直线逼近引出**割线法**(secant method).

假设已知 α 的两个初始猜测, 记为 x_0 和 x_1. 它们可以位于 α 的两侧, 如图 3-3 所示, 或者在 α 的同侧, 如图 3-4 所示. $y = f(x)$ 的图形上两点 $(x_0, f(x_0))$ 和 $(x_1, f(x_1))$ 确定了一条直线, 称为**割线**(secant line). 该直线是 $y = f(x)$ 图形的一种近似, 而它的根 x_2 是 α 的一个近似.

为了推导 x_2 的公式, 我们采用推导牛顿法中使用的类似的方法: 求出直线的方程及它的根 x_2. 该直线的方程是

$$y = p(x) \equiv f(x_1) + (x - x_1) \cdot \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

解方程 $p(x_2) = 0$, 得到

$$x_2 = x_1 - f(x_1) \cdot \frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

求出 x_2 后, 舍弃 x_0, 把 x_1 和 x_2 作为 α 的新的近似值, 这样可得到改进值 x_3. 这个过程可重复进行. 这样, 我们得到一般的迭代公式

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}, \quad n \geqslant 1 \tag{3.27}$$

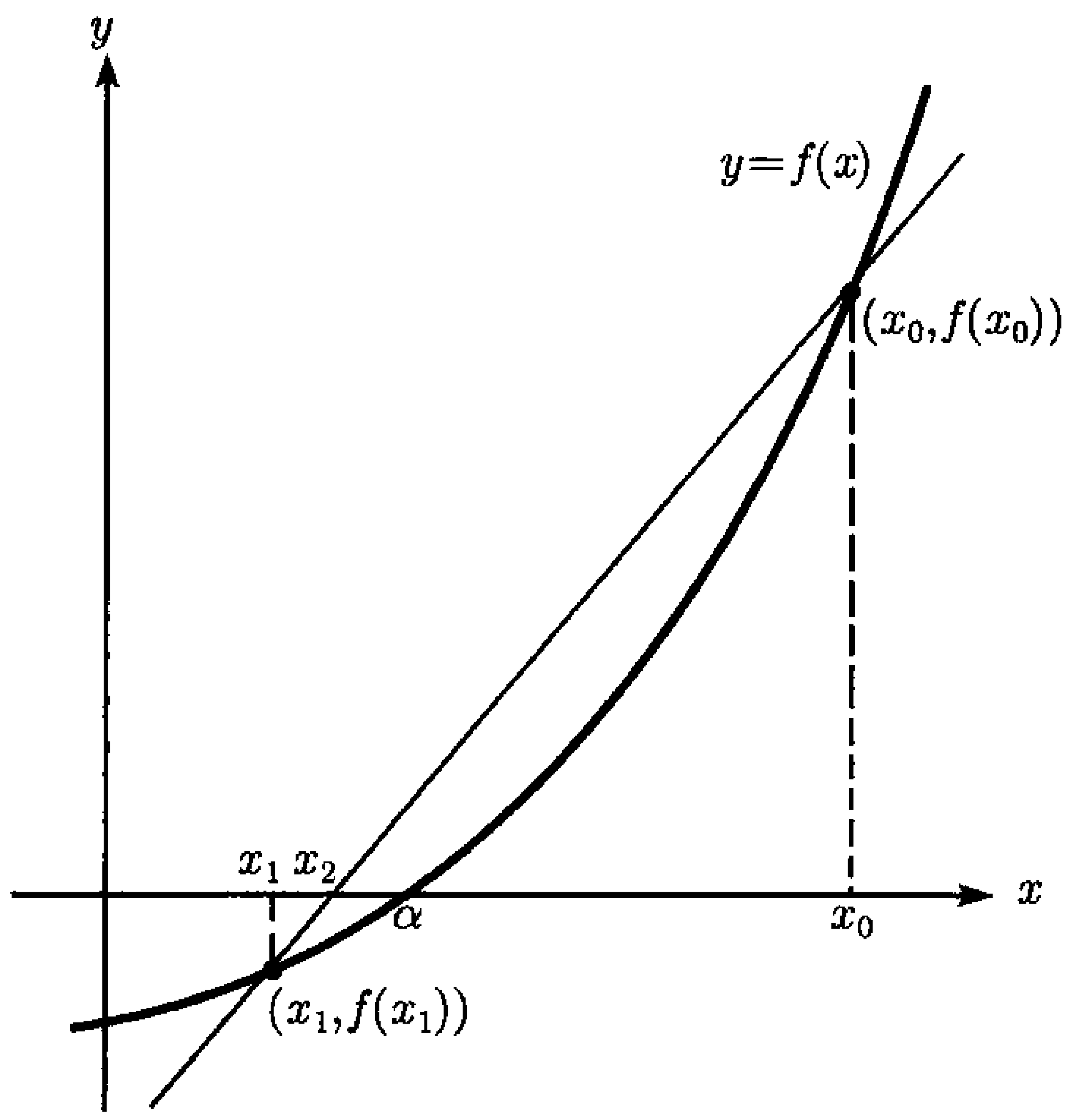


图 3-3　割线法图: $x_1 < \alpha < x_0$

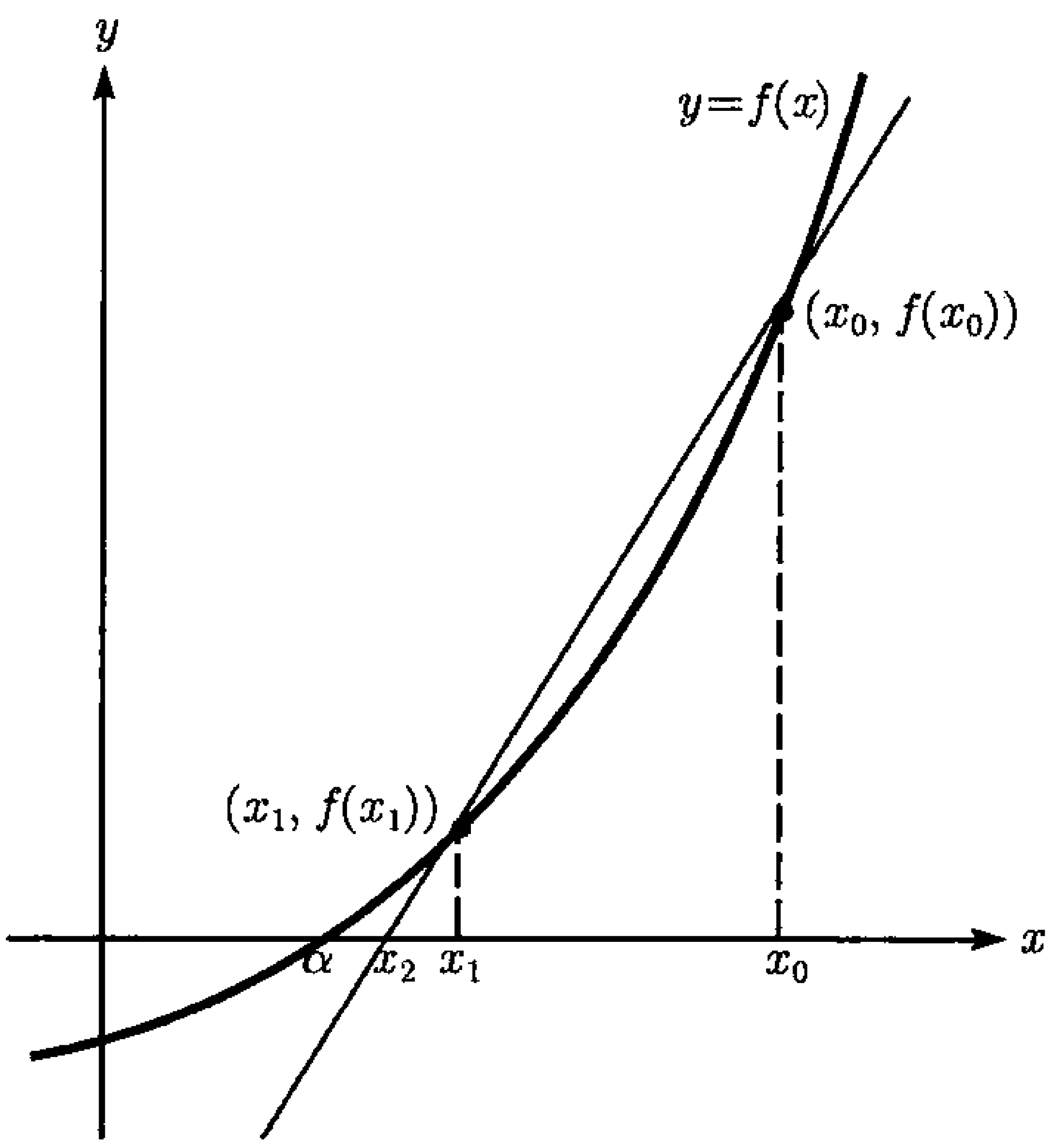


图 3-4　割线法图: $\alpha < x_1 < x_0$

这就是**割线法**(secant method). 由于为了获得一个改进值, 需要两个近似值, 所以它称为两点法. 对分法也是两点法, 但是割线法的收敛速度几乎总是快于对分法.

例 3.3.1　解方程

$$f(x) \equiv x^6 - x - 1 = 0$$

前文中已用这个方程作为使用对分法和牛顿法的例子. 计算结果在表 3-3 中给出, $x_n - x_{n-1}$ 是 $\alpha - x_{n-1}$ 的估计值. 迭代值 x_8 与 α 有 9 位有效数字相同. 与这个方

程的牛顿法 (3.12) 一样, 最初的迭代收敛比较慢. 但当迭代值比较靠近 α 时, 收敛速度就会增加. ■

表 3-3 $x^6 - x - 1 = 0$ 的割线法

n	x_n	$f(x_n)$	$x_n - x_{n-1}$	$\alpha - x_{n-1}$
0	2.0	61.0		
1	1.0	−1.0	−1.0	
2	1.016 129 03	−9.15E−1	1.61E−2	1.35E−1
3	1.190 577 77	6.57E−1	1.74E−1	1.19E−1
4	1.117 655 83	−1.68E−1	−7.29E−2	−5.59E−2
5	1.132 531 55	−2.24E−2	1.49E−2	1.71E−2
6	1.134 816 81	9.54E−4	2.29E−3	2.19E−3
7	1.134 723 65	−5.07E−6	−9.32E−5	−9.27E−5
8	1.134 724 14	−1.13E−9	4.92E−7	4.92E−7

3.3.1 误差分析

经过一些推导, 可以证明 (3.27) 中的迭代 x_n 满足

$$\alpha - x_{n+1} = (\alpha - x_n)(\alpha - x_{n-1})\left[\frac{-f''(\xi_n)}{2f'(\zeta_n)}\right] \tag{3.28}$$

其中未知数 ζ_n 在 x_n 与 x_{n-1} 之间, 未知数 ξ_n 位于三个数 α, x_n 和 x_{n-1} 的最小值和最大值之间. 这个误差公式非常类似于牛顿误差公式 (3.19). 这种情况是预料中的, 因为

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{3.29}$$

所以可以认为割线法其实是牛顿法的一种近似. 容易看出, 在牛顿公式 (3.11) 中使用上述近似值就可得到 (3.27).

公式 (3.28) 可用于进一步的误差分析. 如果选取 x_0 和 x_1 足够靠近 α, 则方法收敛并且

$$\lim_{n\to\infty} \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|^r} = \left|\frac{f''(\alpha)}{2f'(\alpha)}\right|^{r-1} \equiv c \tag{3.30}$$

其中 $r = (\sqrt{5}+1)/2 \doteq 1.62$. 因此, 当 x_n 趋向于 α 时,

$$|\alpha - x_{n+1}| \approx c\,|\alpha - x_n|^{1.62} \tag{3.31}$$

在牛顿法相应的估计式 (3.22) 中, 指数是 2 而不是 1.62. 所以, 牛顿法比割线法收敛得更快. 而且, (3.31) 中的常数 c 与 (3.22) 中的 M 扮演着同样的角色, 它们之间有关系

$$c = |M|^{r-1}$$

牛顿法中对初始猜测的限制 (3.24) 可以用割线法中类似的条件代替, 但是我们在此不做进一步的讨论. 当迭代值 x_n 足够靠近这个根时, 可用 (3.31) 来证明误差估计

$$\alpha - x_{n-1} \approx x_n - x_{n-1} \tag{3.32}$$

是合理的.

例 3.3.2 关于表 3-3 中迭代 x_5, 有

$$\alpha - x_5 \doteq 2.19\mathrm{E}-3$$

$$x_6 - x_5 \doteq 2.29\mathrm{E}-3 \qquad ■(3.33)$$

MATLAB 程序: 割线法的实施 程序secant用来实现割线法, 它包括误差估计和预防无限循环的措施. 输入参数max_iterate是计算的迭代次数的上界, 这个参数可以防止无限循环现象.

```
function root = secant(x0,x1,error_bd,max_iterate)
%
% function secant(x0,x1,error_bd,max_iterate)
%
% This implements the secant method for solving an
% equation f(x) = 0.
%
% The parameter error_bd is used in the error test for the
% accuracy of each iterate.  The parameter max_iterate is
% an upper limit on the number of iterates to be computed.
% Two initial guesses, x0 and x1, must also be given.
%
% For the given function f(x), an example of a calling
% sequence might be the following:
%       root = secant(x0,x1,1.0E-12,10)
% The function f(x) is given below.
%
% The program prints the iteration values
%       iterate_number, x, f(x), error
% The value of x is the most current initial guess, called
% previous_iterate here, and it is updated with each
% iteration.  The value of error is
%       error = newly_computed_iterate - previous_iterate
% and it is an estimated error for previous_iterate.
% Tap the carriage return to continue with the iterati

format short e

error = 1;
fx0 = f(x0);
```

```
it_count = 0;
iteration = [it_count x0 fx0]

while abs(error) > error_bd & it_count <= max_iterate
  it_count = it_count + 1;
  fx1 = f(x1);
  if fx1 - fx0 == 0
    disp('f(x1) = f(x0); Division by zero; Stop')
    return
  end
  x2 = x1 - fx1*(x1-x0)/(fx1-fx0);
  error = x2 - x1;
% Internal print of secant method.  Tap the carriage
% return key to continue the computation.
  iteration = [it_count x1 fx1 error]
  pause
  x0 = x1;
  x1 = x2;
  fx0 = fx1;
end

if it_count > max_iterate
  disp('The number of iterates calculated exceeded')
  disp('max_iterate.  An accurate root was not')
  disp('calculated.')
else
  format long
  root = x2
  format short
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = f(x)
%
% function to define equation for rootfinding problems.
%
value = x.^6 - x - 1;
```

3.3.2 牛顿法和割线法的比较

由前面的讨论知, 牛顿法的收敛速度要比割线法的收敛速度快. 因此, 牛顿法只需要较少的迭代次数便可达到给定的误差容限. 然而, 牛顿法每次迭代需要两次函数求值, 即计算 $f(x_n)$ 和 $f'(x_n)$. 如果保留从前一次迭代得到的 $f(x_{n-1})$ 的值, 则割线法的每次迭代只需一次计算 $f(x_n)$ 的值. 因而, 每次迭代割线法比牛顿法省时.

到底使用哪一种方法依赖于上面讨论过的因素, 包括计算 $f'(x)$ 的难度和计算量, 也可能包括方法使用的方便程度. 对于编程和理解而言, 牛顿法十分简单; 但是对于具有复杂 $f'(x)$ 的许多问题, 在计算机上实际运行时间方面, 割线法很可能更快一些.

注 牛顿法和割线法的推导利用了数值分析的一个一般性原则. 在试图求解一个问题时, 如果没有直接或者简单的求解方法, 就用另一个较容易求解的问题来逼近原问题. 两种数值方法都用非常容易的线性方程求根问题的解代替 $f(x) = 0$ 的解. 这是 1.1 节末尾给出的一般准则 (1.10) 的另一个例子. 我们也可按照所使用的逼近特性给出下面的评注.

> **一般准则**
> 当处理含有可微函数 $f(x)$ 的问题时, 用线性函数逼近 $f(x)$, 得到一个相近的问题. (3.34)

数学问题的线性化在应用数学和数值分析中是很常见的.

3.3.3 MATLAB 函数 fzero

MATLAB 包含一个求根程序`fzero`, 它使用了对分法和割线法的思想. 如同许多 MATLAB 程序那样, 有几种可能的使用序列. 指令

$$\texttt{root} = \texttt{fzero}(\texttt{f_name}, [\texttt{a}, \texttt{b}])$$

产生 $[a, b]$ 中的一个根, 这里假设 $f(a)f(b) \leqslant 0$. 指令

$$\texttt{root} = \texttt{fzero}(\texttt{f_name}, \texttt{x0})$$

找出函数在 x0 附近的一个根, 其中的误差容限是机器的最大精度. 用户也可以给出自己的误差容限. 这是一个极好的求根程序, 确保收敛并有高的效率.

习题

1. 利用割线法, 求 3.1 节习题 1 中方程的根, 误差容限为 $\varepsilon = 10^{-6}$.
2. 考虑方程 (3.2), 取 $P_{\text{in}} = 1\ 000$, $P_{\text{out}} = 20\ 000$, $N_{\text{in}} = 30$, $N_{\text{out}} = 20$, 用 $\varepsilon = 0.000\ 1$ 的精度求 r.
3. 利用割线法, 解 3.2 节习题 6. 初值可取 $x_0 = -1$ 及 $x_1 = 0$.
4. 继续例题 3.3.1, 用实验证实误差估计 (3.31). 为此, 首先利用公式 (3.30) 计算常数 c. 然后对 $n = 2, 3, \cdots, 7$, 计算和比较误差估计 (3.31) 的两端.
5. 利用割线法, 重做 3.1 节习题 7 和 3.2 节习题 10.
6. 利用割线法, 重做 3.2 节习题 11.
7. 作为 (3.28) 的证明过程的一部分, 用代数技巧证明

$$\alpha - x_{n+1} = -(\alpha - x_n)(\alpha - x_{n-1})\frac{f[x_{n-1}, x_n, \alpha]}{f[x_{n-1}, x_n]}$$

其中

$$f[a,b] = \frac{f(b) - f(a)}{b - a}$$

$$f[a,b,c] = \frac{f[b,c] - f[a,b]}{c - a}$$

它们称为**牛顿差商**(Newton divided difference), 在 4.1 节中将对其进行讨论. 把公式 (4.24) 用于上述误差公式便可推导出 (3.28).

8. 把公式 (3.28) 写成

$$\alpha - x_{n+1} \approx M(\alpha - x_n)(\alpha - x_{n-1}), \quad n \geqslant 0$$

其中 M 是在 (3.21) 中定义的. 上式两端乘以 M, 得

$$|M(\alpha - x_{n+1})| \approx |M(\alpha - x_n)|\,|M(\alpha - x_{n-1})|$$

设 $B_n = |M(\alpha - x_n)|$, $n \geqslant 0$. 要使得 x_n 收敛于 α, B_n 必须收敛于 0. 上述公式变为

$$B_{n+1} \approx B_n B_{n-1}$$

为了简单起见, 假设 $B_0 = B_1 = \delta$.

(a) 计算用 δ 表示的 $B_2, B_3, B_4, B_5, B_6, B_7$ 的近似值.

(b) 如果记 $B_n = \delta^{q_n}$, $n \geqslant 0$, 给出用 q_{n-1} 和 q_n 表示的 q_{n+1} 的一个公式. 试问 q_0 和 q_1 是什么?

(c) 对于较大的 n, 例如 $n \geqslant 4$, 用实验确认

$$q_n \approx \frac{1}{\sqrt{5}} r^{n+1} \doteq \frac{1}{\sqrt{5}}(1.618)^{n+1}$$

其中的数 r 在 (3.30) 中出现过. 数 $\{q_n\}$ 称为**斐波那契数**①(Fibonacci number), 数 r 称为**黄金分割**(golden mean)(有些作者把 $r-1$ 定义为黄金分割). 有关这方面更多的细节, 参看阿特金森 (Atkinson, 1989, 第 68~69 页).

(d) 用 (c) 证明

$$\frac{B_{n+1}}{B_n^r}$$

趋向于一个常数. 求出这个常数. 此结论可用来构造 (3.30) 的一个证明.

3.4 不动点迭代

牛顿法 (3.11) 和割线法 (3.27) 分别是单点迭代法和两点迭代法的例子. 本节更一般地介绍迭代法, 给出单点迭代公式的一般理论.

① 斐波那契数也称为黄金分割数. —— 译者注

作为一个启发性的例子, 考虑求解方程

$$x^2 - 5 = 0 \tag{3.35}$$

其根 $\alpha = \sqrt{5} \doteq 2.236\,1$. 我们给出求解这个方程的四种迭代法.

I1. $x_{n+1} = 5 + x_n - x_n^2$

I2. $x_{n+1} = \frac{5}{x_n}$

I3. $x_{n+1} = 1 + x_n - \frac{1}{5}x_n^2$

I4. $x_{n+1} = \frac{1}{2}\left(x_n + \frac{5}{x_n}\right)$

这四种迭代有一个共同性质：如果序列 $\{x_n|n \geqslant 0\}$ 有极限 α, 则 α 是 (3.35) 的根. 对每个方程, 上述性质可如下验证：用 α 替换 x_n 和 x_{n+1}, 然后证明这可推出 $\alpha = \pm\sqrt{5}$. 表 3-4 给出了这四种迭代法的迭代值 x_n. 为了解释这些数值结果, 我们介绍单点迭代公式的一般理论.

表 3-4 迭代 I1 到 I4

n	x_n:I1	x_n:I2	x_n:I3	x_n:I4
0	2.5	2.5	2.5	2.5
1	1.25	2.0	2.25	2.25
2	4.687 5	2.5	2.237 5	2.236 1
3	−12.285 2	2.0	2.236 2	2.236 1

对于适当定义的连续函数 $g(x)$, 迭代 I1 到 I4 都具有下列形式

$$x_{n+1} = g(x_n) \tag{3.36}$$

例如, 就 I1 来说, $g(x) = 5 + x - x^2$. 如果迭代序列 x_n 收敛到一点 α, 则

$$\lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} g(x_n)$$

$$\alpha = g(\alpha)$$

因此, α 是方程 $x = g(x)$ 的解. 我们称 α 为函数 g 的一个*不动点*(fixed point).

本节给出一般理论来解释迭代法 $x_{n+1} = g(x_n)$ 何时收敛于 g 的不动点. 我们从 $x = g(x)$ 解的存在性引理开始讨论.

引理 3.4.1 设 $g(x)$ 是区间 $[a,b]$ 上的连续函数, 并假设 g 满足下列性质

$$a \leqslant x \leqslant b \quad \Longrightarrow \quad a \leqslant g(x) \leqslant b \tag{3.37}$$

则方程 $x = g(x)$ 在区间 $[a,b]$ 内至少有一个解 α.

证明 定义函数 $f(x) = x - g(x)$. 对于 $a \leqslant x \leqslant b$, 它是一个连续函数. 此外, $f(a) \leqslant 0$ 并且 $f(b) \geqslant 0$. 根据介值定理 (见附录 A), 必定存在 $[a,b]$ 中的一点 x 使得 $f(x) = 0$. 通常我们用 α 表示 x 的这个值. ■

方程 $x = g(x)$ 的解的图形解释见图 3-5. 解 α 是 $y = x$ 和 $y = g(x)$ 的图形交点的 x 坐标.

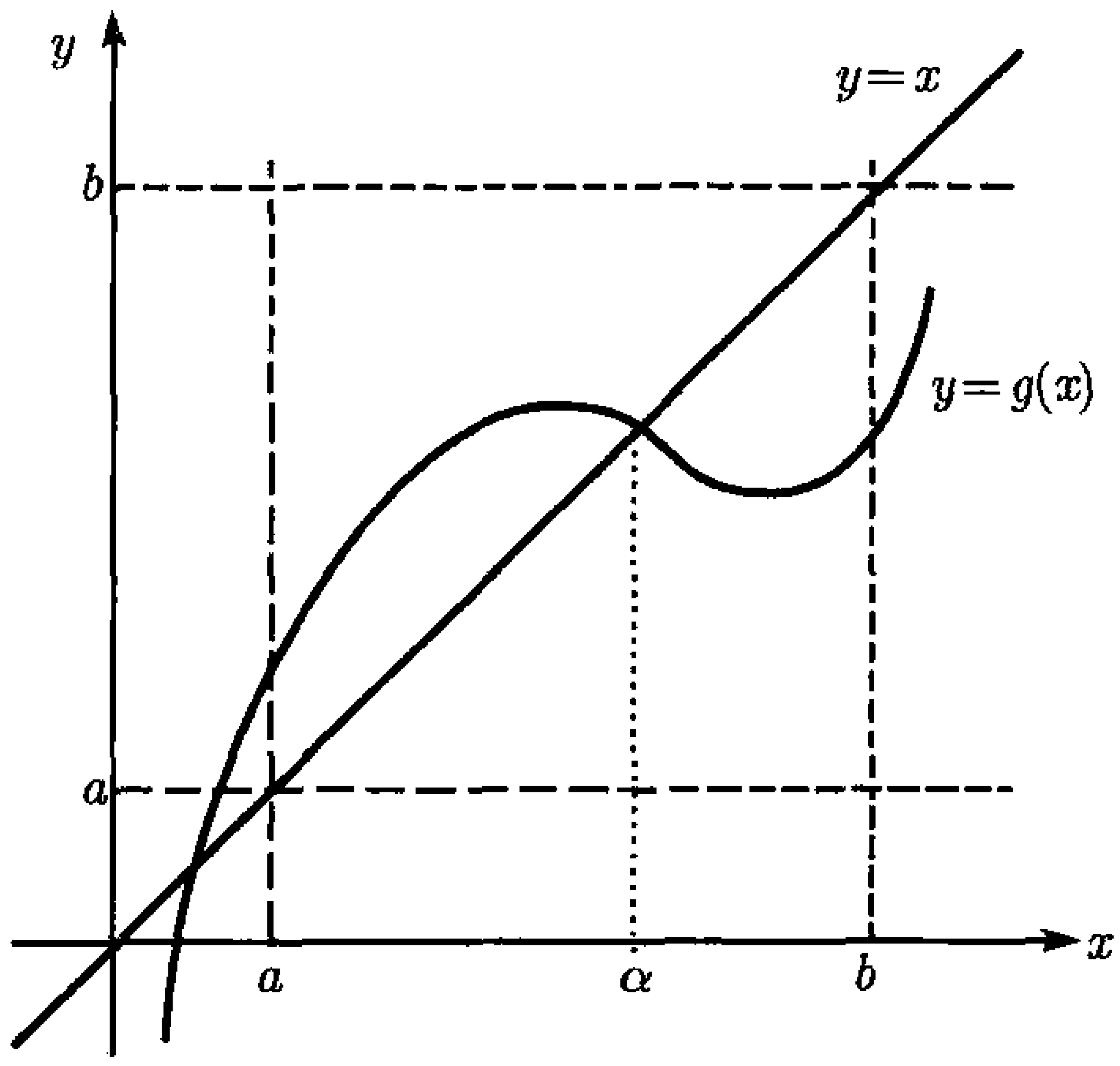


图 3-5 引理 3.4.1 的例子

定理 3.4.2(压缩映射定理) 假设对于 $a \leqslant x \leqslant b$, $g(x)$ 和 $g'(x)$ 连续, 并且 g 满足 (3.37). 再假设

$$\lambda \equiv \max_{a \leqslant x \leqslant b} |g'(x)| < 1 \tag{3.38}$$

则

S1. 方程 $x = g(x)$ 在区间 $[a,b]$ 中有唯一解 α.

S2. 对 $[a,b]$ 中任意选取的初始估计 x_0 , 迭代 x_n 将收敛于 α.

S3.

$$|\alpha - x_n| \leqslant \frac{\lambda^n}{1-\lambda}|x_0 - x_1|, \quad n \geqslant 0 \tag{3.39}$$

S4.

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\alpha) \tag{3.40}$$

从而对于靠近 α 的 x_n, 有

$$\alpha - x_{n+1} \approx g'(\alpha)(\alpha - x_n) \tag{3.41}$$

证明 因为证明过程提供一些有用的信息, 所以我们给出具体的细节. 首先注意, 对 g 的假设允许使用引理 3.4.1 来断言 $x = g(x)$ 至少存在一个解. 另外, 利用

中值定理 (见附录 A), 对 $[a,b]$ 中任意两点 w 和 z, 有

$$g(w)-g(z)=g'(c)(w-z)$$

其中 c 在 w 和 z 之间. 利用 (3.38), 得到

$$|g(w)-g(z)|=|g'(c)|\,|w-z|\leqslant\lambda\,|w-z|\,,\qquad a\leqslant w,z\leqslant b \tag{3.42}$$

S1. 假设方程 $x=g(x)$ 有两个解, 记为 α 和 β. 则 $\alpha=g(\alpha)$ 和 $\beta=g(\beta)$. 把它们相减, 得到

$$\alpha-\beta=g(\alpha)-g(\beta)$$

取绝对值并利用 (3.42):

$$|\alpha-\beta|\leqslant\lambda\,|\alpha-\beta|\,,\quad(1-\lambda)\,|\alpha-\beta|\leqslant 0$$

因为 $\lambda<1$, 所以必有 $\alpha=\beta$; 因此, 方程 $x=g(x)$ 在区间 $[a,b]$ 中只有一个解.

S2. 根据假设 (3.37), 可以证明, 对 $[a,b]$ 中任一初始猜测 x_0, 迭代值 x_n 都将保持在 $[a,b]$ 中. 例如, 如果 $a\leqslant x_0\leqslant b$, 则由 (3.37) 可推出 $a\leqslant g(x_0)\leqslant b$. 因为 $x_1=g(x_0)$, 这表明 x_1 在 $[a,b]$ 中. 重复上述的论证可知所有的 x_n 都在区间 $[a,b]$ 中.

为了证明迭代序列收敛, 从 $\alpha=g(\alpha)$ 减去 $x_{n+1}=g(x_n)$, 得到

$$\alpha-x_{n+1}=g(\alpha)-g(x_n)=g'(c_n)(\alpha-x_n) \tag{3.43}$$

其中 c_n 在 w 和 z 之间. 利用假设 (3.38), 有

$$|\alpha-x_{n+1}|\leqslant\lambda\,|\alpha-x_n|\,,\quad n\geqslant 0 \tag{3.44}$$

由此, 我们可用归纳法得到

$$|\alpha-x_n|\leqslant\lambda^n\,|\alpha-x_0|\,,\quad n\geqslant 0 \tag{3.45}$$

因为 $\lambda<1$, 所以当 $n\to\infty$ 时 (3.45) 右端趋向于零. 这就证明了当 $n\to\infty$ 时, $x_n\to\alpha$.

S3. 利用 (3.44) 与 $n=1$, 得到

$$\begin{aligned}|\alpha-x_0|&\leqslant|\alpha-x_1|+|x_1-x_0|\leqslant\lambda\,|\alpha-x_0|+|x_1-x_0|\\(1-\lambda)\,|\alpha-x_0|&\leqslant|x_1-x_0|\\|\alpha-x_0|&\leqslant\frac{1}{1-\lambda}\,|x_1-x_0|\end{aligned} \tag{3.46}$$

把该式与 (3.45) 结合, 可推导出不等式 (3.39).

S4. 利用 (3.43) 可写出

$$\lim_{n\to\infty}\frac{\alpha - x_{n+1}}{\alpha - x_n} = \lim_{n\to\infty} g'(c_n)$$

由 S2 知, 每个 c_n 在 α 与 x_n 之间, 而且 $x_n \to \alpha$. 因此, $c_n \to \alpha$. 再结合函数 $g'(x)$ 的连续性, 得到

$$\lim_{n\to\infty} g'(c_n) = g'(\alpha)$$

这样就证明了 (3.40). ■

我们需要一种更精确的方式去论述迭代法收敛速度的概念. 如果对于常数 $c \geqslant 0$ 和 $p \geqslant 1$,

$$|\alpha - x_{n+1}| \leqslant c\,|\alpha - x_n|^p, \quad n \geqslant 0$$

则称序列 $\{x_n | n \geqslant 0\}$ 以 p 阶收敛于 α. 序列 $\{x_n | n \geqslant 0\}$ 的**收敛阶**(order of convergence) 往往指使得上面的不等式成立的最大可能的 p. $p=1, p=2$ 和 $p=3$ 的情形分别被称为**线性收敛**(linear convergence)、**二次收敛**(quadratic convergence) 和**三次收敛**(cubic convergence). 牛顿法通常二次收敛; 割线法的收敛阶是 $p=(1+\sqrt{5})/2$. 对线性收敛序列, 我们要求附加条件 $c<1$; 否则, 误差 $\alpha - x_n$ 不一定收敛于零.

在定理 3.4.2 中, 如果 $|g'(\alpha)| < 1$, 则不等式 (3.44) 表明迭代序列 x_n 线性收敛. 如果还知道 $g'(\alpha) \neq 0$, 则关系式 (3.41) 表明收敛正好是线性的, 而且没有更高阶收敛的可能性. 在这种情况下, 我们把数值 $|g'(\alpha)|$ 称为线性收敛速率.

实际上, 很少能直接使用定理 3.4.2. 主要原因在于很难找到满足 (3.37) 的区间 $[a,b]$. 作为替代, 我们从该定理出发来推导出一种实用的形式. 注意到, 关键是关系式 (3.43), 它表明当迭代 x_n 靠近 α 时迭代误差的情况.

推论 3.4.3 假设对于某个区间 $c<x<d$, $g(x)$ 和 $g'(x)$ 连续, 并且不动点 α 包含在这个区间内. 此外, 假设

$$|g'(\alpha)| < 1 \tag{3.47}$$

则存在 α 附近的一个区间 $[a,b]$, 使得定理 3.4.2 的假设成立, 因此定理的结论也成立. 相反地, 如果 $|g'(\alpha)| > 1$, 则迭代法 $x_{n+1} = g(x_n)$ 不收敛于 α. [当 $|g'(\alpha)| = 1$ 时, 迭代法可能收敛, 也可能不收敛; 此时, 即使收敛, 收敛速度也会非常缓慢.]

这个推论的证明留在习题 10 中处理. 我们可以利用推论 3.4.3 来讨论迭代法 I1 到 I4 的收敛性. 记住 $\alpha = \sqrt{5}$.

I1. $g(x) = 5 + x - x^2$, $\quad g'(x) = 1 - 2x$, $\quad g'(\alpha) = 1 - 2\sqrt{5} < -1$. 因而, 迭代 I1 不收敛于 $\sqrt{5}$.

I2. $g(x) = 5/x, \quad g'(x) = -5/x^2, \quad g'(\alpha) = -1$. 我们不能推断迭代收敛或者发散. 但是根据表 3-4, 迭代显然不收敛于 α.

I3. $g(x) = 1 + x - \frac{1}{5}x^2, \quad g'(x) = 1 - \frac{2}{5}x, \quad g'(\alpha) = 1 - \frac{2}{5}\sqrt{5} \doteq 0.106$. 根据推论, 迭代序列收敛. 根据 (3.40), 当 x_n 靠近 α 时, 有

$$|\alpha - x_{n+1}| \approx 0.106\,|\alpha - x_n|$$

即每进行一次迭代, 误差大约减少 90%.

I4. $g(x) = \frac{1}{2}(x + 5/x), \quad g'(x) = \frac{1}{2}(1 - 5/x^2), \quad g'(\alpha) = 0$. 从而, 收敛条件满足. 注意到, 这实际上就是计算 $\sqrt{5}$ 的牛顿法.

通常很难知道如何把求根问题 $f(x) = 0$ 转换为可导致收敛迭代方法的不动点问题 $x = g(x)$. 利用推论 3.4.3, 习题 9 给出了一个这样的过程.

根据 $g'(\alpha)$ 的值的大小, 不动点迭代 x_n 会出现不同的情况, 如图 3-6 所示. 先看收敛性, 考虑 $x_1 = g(x_0)$ 的情形, 它是 $y = g(x)$ 的图形在 x_0 点的高. 用直线 $y = x$ 和高 $y = x_1$ 把数 x_1 带回到 x 轴. 对每次迭代继续这个过程, 当 $g'(\alpha) > 0$ 时得到一种梯级般升降的情况. 当 $g'(\alpha) < 0$ 时, 迭代序列围绕不动点 α 摆动, 如图 3-6 中看到的那样.

例 3.4.4 表 3-5 给出了迭代法 I3 的一些数值结果, 包括迭代收敛性方面的一些信息, 如误差及比率

表 3-5 迭代 $x_{n+1} = 1 + x_n - \frac{1}{5}x_n^2$

n	x_n	$\alpha - x_n$	r_n
0	2.5	−2.64E−1	
1	2.25	−1.39E−2	0.052 8
2	2.237 5	−1.43E−3	0.102 8
3	2.236 218 75	−1.51E−4	0.105 3
4	2.236 083 89	−1.59E−5	0.105 5
5	2.236 069 66	−1.68E−6	0.105 6
6	2.236 068 15	−1.77E−7	0.105 6
7	2.236 068 00	−1.87E−8	0.105 6

$$r_n = \frac{\alpha - x_n}{\alpha - x_{n-1}} \tag{3.48}$$

数值上, r_n 的值收敛于 $g'(\alpha) \doteq 0.105\,573$, 这与 (3.40) 一致. ■

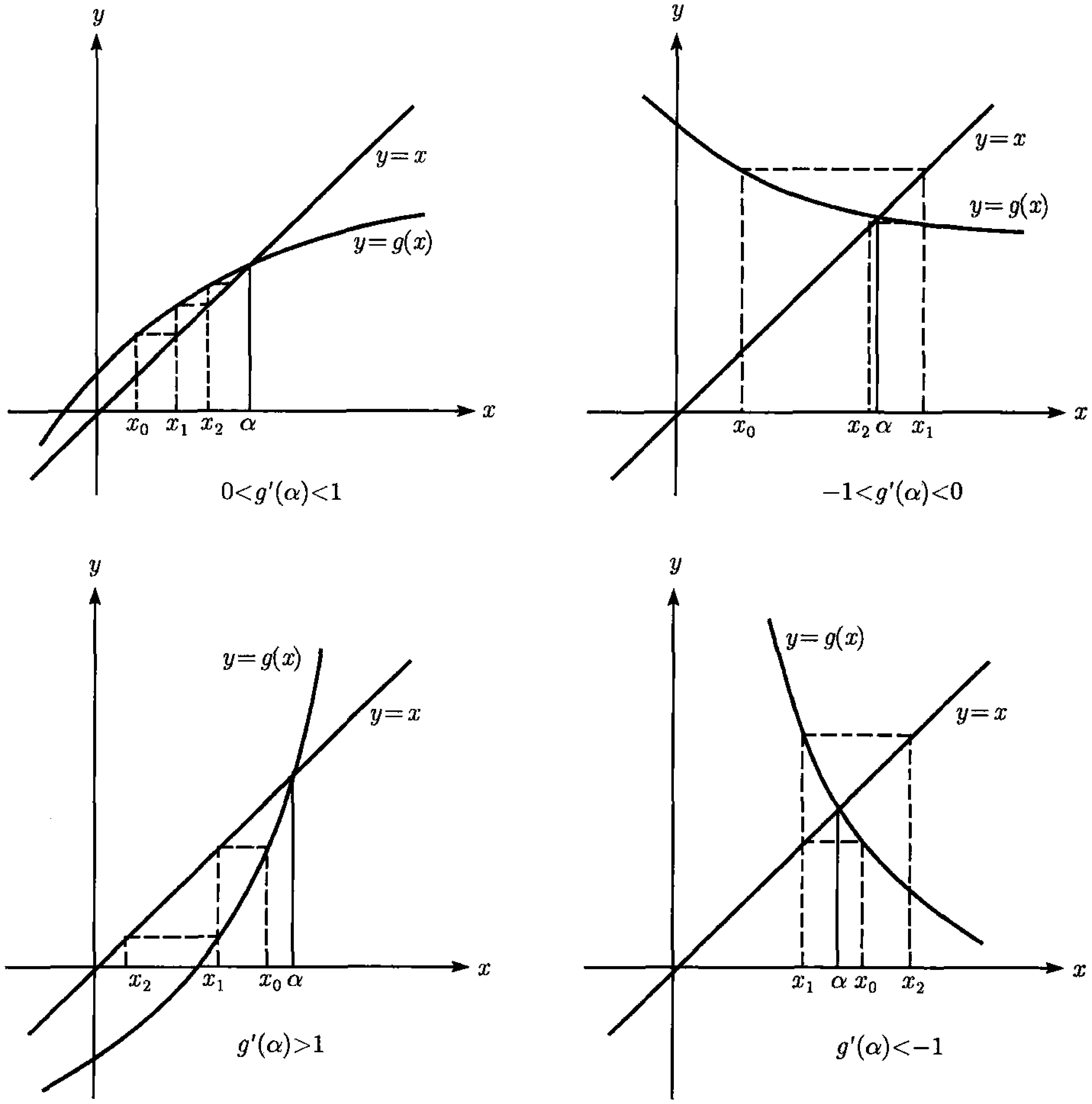


图 3-6 收敛及不收敛序列 $x_{n+1} = g(x_n)$ 的图解

3.4.1 艾特肯误差估计和外推

利用公式 (3.41), 可以估计迭代值 x_n 的误差以及加速迭代收敛. 用 λ 表示 $g'(\alpha)$. 考虑使 (3.41) 成立的那些指标. 用 $n-1$ 代替 n 把它写为

$$\alpha - x_n \approx \lambda(\alpha - x_{n-1}) \tag{3.49}$$

求解 α 并且写成便于计算的形式,

$$\alpha \approx x_n + \frac{\lambda}{1-\lambda}(x_n - x_{n-1}) \tag{3.50}$$

我们来估计 λ 的值. 由于不知道真解 α, 所以不能从 λ 的定义出发来计算它. 对于 (3.48) 中的比率, 情况相同. 为了估计 λ, 利用比率

$$\lambda_n = \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}, \quad n \geqslant 2 \tag{3.51}$$

利用中值定理, 我们有

$$\lambda_n = \frac{g(x_{n-1}) - g(x_{n-2})}{x_{n-1} - x_{n-2}} = g'(c_n)$$

其中 c_n 满足 $\min\{x_{n-1}, x_{n-2}\} \leqslant c_n \leqslant \max\{x_{n-1}, x_{n-2}\}$. 因为当 $n \to \infty$ 时 $x_n \to \alpha$, 所以 $c_n \to \alpha$; 根据 $g'(x)$ 的连续性, $g'(c_n) \to g'(\alpha) = \lambda$. 因而, 当 $n \to \infty$ 时 $\lambda_n \to \lambda$. 所以, 当 $n \to \infty$ 时, 由 (3.51) 定义的比率将是 λ 的一个越来越好的估计. 结合 (3.50) 和 (3.51), 得到

$$\alpha \approx x_n + \frac{\lambda_n}{1 - \lambda_n}(x_n - x_{n-1}) \tag{3.52}$$

此式称为**艾特肯外推公式**(Aitken's extrapolation formula). 它的等价形式

$$\alpha - x_n \approx \frac{\lambda_n}{1 - \lambda_n}(x_n - x_{n-1}) \tag{3.53}$$

给出了**艾特肯误差估计**(Aitken's error estimate). 此式可用来估计原来迭代 $\{x_n\}$ 的误差, 也可用公式 (3.52) 来产生一个收敛更快的序列.

例 3.4.5 重复关于迭代法 I3 的例子. 表 3-6 中包含差 $x_n - x_{n-1}$、比率 λ_n 以及 (3.53) 产生的估计误差. 请读者比较表中的估计误差与表 3-5 中给出的实际误差. ■

表 3-6 迭代 $x_{n+1} = 1 + x_n - \frac{1}{5}x_n^2$ 和艾特肯误差估计

n	x_n	$x_n - x_{n-1}$	λ_n	误差估计
0	2.5			
1	2.25	−2.50E−1		
2	2.237 5	−1.25E−2	0.050 0	−6.58E−4
3	2.236 218 75	−1.28E−3	0.102 5	−1.46E−4
4	2.236 083 89	−1.35E−4	0.105 3	−1.59E−5
5	2.236 069 66	−1.42E−5	0.105 5	−1.68E−6
6	2.236 068 15	−1.50E−6	0.105 6	−1.77E−7
7	2.236 068 00	−1.59E−7	0.105 6	−1.87E−8

注 由于很多原因, 我们有必要进行数值方法的理论误差分析. 数值方法有时可以令人满意地运行, 有时则不行, 甚至完全不能运行, 这就需要很好地理解数值方法. 数学证明能让我们确信在所作的假设下数值方法本身的正确性. 对方法的理解还能帮助我们改进数值方法的性能. 应用定理 3.4.2 得到艾特肯外推公式 (3.52) 这一过程印证了下列结论.

> **一般准则**
>
> 通过了解数值方法中误差的情况, 常常可以改进这个方法并且得到另一个收敛更快的方法. (3.54)

我们将在本书其他章节举例说明这一点.

3.4.2 高阶迭代公式

当 $g'(\alpha) = 0$ 时, 收敛性很好, 但收敛关系式 (3.41) 不太有用. 在这种情况下, 为了改进定理 3.4.2, 我们假设 $g(x)$ 二次连续可微. 考虑 $g(x_n)$ 关于 α 的泰勒展开式:

$$g(x_n) = g(\alpha) + (x_n - \alpha)g'(\alpha) + \frac{1}{2}(x_n - \alpha)^2 g''(c_n) \tag{3.55}$$

c_n 在 x_n 和 α 之间. 利用 $x_{n+1} = g(x_n), \alpha = g(\alpha)$和$g'(\alpha) = 0$, 我们得到

$$x_{n+1} = \alpha + \frac{1}{2}(x_n - \alpha)^2 g''(c_n)$$

$$\alpha - x_{n+1} = -\frac{1}{2}(\alpha - x_n)^2 g''(c_n) \tag{3.56}$$

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{1}{2}g''(\alpha) \tag{3.57}$$

如果 $g''(\alpha) \neq 0$, 则这个公式表明, 迭代法 $x_{n+1} = g(x_n)$ 是 2 阶 (order 2) 的, 或者说*二次收敛的*(quadratically convergent).

如果还有 $g''(\alpha) = 0$, 也许某些高阶导数在 α 处同样为零, 则可代替 (3.55) 以高阶的泰勒展开, 直到最后的误差项含有在 α 处不为零的 $g(x)$ 的导数. 这样的数值方法收敛阶大于 2.

例如, 把牛顿法看作一个不动点迭代:

$$x_{n+1} = g(x_n), \quad g(x) = x - \frac{f(x)}{f'(x)} \tag{3.58}$$

我们有

$$g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$$

如果 $f'(\alpha) \neq 0$, 则

$$g'(\alpha) = 0$$

如果还有 $f''(\alpha) \neq 0$, 则可以证明 $g''(\alpha) \neq 0$. 这样, 如果 $f'(\alpha) \neq 0$ 而且 $f''(\alpha) \neq 0$, 则根据 (3.57), 上述结果表明牛顿法是 2 阶的.

习题

1. (a) 用 $x_0=1$, 计算下列迭代的前六项

$$x_{n+1}=1+0.3\sin(x_n)$$

选择其他初值 x_0, 重复上述计算.

(b) 求一个区间 $[a,b]$, 使其满足定理 3.4.2 的假设条件.

提示: 对于 $g(x)=1+0.3\sin(x)$, 取

$$a=\min_{-\infty<x<\infty} g(x), \quad b=\max_{-\infty<x<\infty} g(x)$$

(c) 根据前面的讨论过程作一个与表 3-5 同样格式的表格. 真解是 $\alpha \doteq 1.288\,091\,313\,211\,84$.

2. 对迭代 $x_{n+1}=0.5/(1+x_n^2)$ 重复习题 1. 真解是 $\alpha \doteq 0.423\,853\,799\,069\,783$.
3. 方程 $x=\mathrm{e}^{-x}$ 有多少解? 对于适当选取的 x_0 迭代 $x_{n+1}=\mathrm{e}^{-x_n}$ 收敛吗? 当 $x_0=0$ 时计算迭代的前六项.
4. 对迭代 $x_{n+1}=1+\tan^{-1}(x_n)$ 重复习题 3.
5. 对任何常数 c 和 d, $|d|<1$, 证明方程 $x=c+d\cos(x)\equiv g(x)$ 有唯一解 α. 证明迭代 $x_{n+1}=c+d\cos(x_n)$ 收敛于 α, 并界定其收敛速度.
6. 把方程 $x^2-5=0$ 转换为不动点问题

$$x=x+c(x^2-5)\equiv g(x)$$

其中 c 是非零常数. 确定 c 可能取的值以保证迭代

$$x_{n+1}=x_n+c(x_n^2-5)$$

收敛到 $\alpha=\sqrt{5}$.

7. 如果方程 $x=\sqrt{1+x}$ 有解的话, 它的解 α 是什么? 迭代 $x_{n+1}=\sqrt{1+x_n}$ 会收敛于某个解吗 (假设选取 x_0 充分靠近 α)?
8. 倘若选取 x_0 充分靠近 α, 下列哪个迭代会收敛于指定的 α? 如果收敛, 确定收敛阶.

(a) $x_{n+1}=\frac{15x_n^2-24x_n+13}{4x_n}, \quad \alpha=1.$

(b) $x_{n+1}=\frac{3}{4}x_n+1/x_n^3, \quad \alpha=\sqrt{2}.$

9. 考虑求根问题 $f(x)=0$, 其根为 α 且 $f'(x)\neq 0$. 把它转换成不动点问题

$$x=x+cf(x)\equiv g(x)$$

其中 c 是非零常数. 应该如何选择 c 以保证迭代

$$x_{n+1}=x_n+cf(x_n)$$

快速收敛到 α(倘若选取 x_0 充分靠近 α)? 把你选取 c 的方法应用到求根问题 $x^3-5=0$.

10. 证明推论 3.4.3. 为此, 首先注意结合 $|g'(\alpha)|<1$ 和 $g'(x)$ 的连续性, 对某个 $r>0$, 证明

$$|\alpha - x| \leqslant r \text{ 可推出 } |g'(\alpha)-g'(x)| \leqslant \frac{1-|g'(\alpha)|}{2}$$

考虑区间 $[a,b]=[\alpha-r,\alpha+r]$ 并在该区间上界定 $g'(x)$ 的大小范围.

11. 对某些 c 的值, 迭代 $x_{n+1}=2-(1+c)x_n+cx_n^3$ 将收敛于 $\alpha=1$(假设选取 x_0 充分靠近 α). 求出使这个迭代收敛的 c 的值. 如果存在的话, c 为何值时收敛是二次的?

12. 考虑方程 $x=g_c(x)\equiv cx(1-x)$, 其中 c 是非零常数. 这个方程有两个解, 用 α_c 表示它的非零解. α_c 是多少? c 取何值时迭代 $x_{n+1}=g_c(x_n)$ 收敛到 α_c(假设选取 x_0 充分靠近 α)?

注: 方程 $x=g_c(x)$ 称为**逻辑斯谛方程**(logistic equation). 作为**混沌数学理论**(mathematical theory of chaos) 中的例子, 这个方程和相应的迭代 $x_{n+1}=g_c(x_n)$ 近年来引起了人们的极大兴趣. 为了观察到一些有趣的现象, 缓慢地增加 c 的值, 使其经过本题前面求出的区间, 保持 $c<4$. 观察 n 的值很大时的迭代情况. 更多的讨论, 参见伊恩 · 斯图尔特所著《上帝掷骰子吗?》一书 (Ian Stewart, Does God Play Dice?, Blackwell Ltd. Publishers, Oxford, 1989, P.155).

13. 用艾特肯误差估计公式 (3.53) 估计下列迭代中的误差 $\alpha-x_2$.

(a) $x_{n+1}=\mathrm{e}^{-x_n}, \quad x_0=0.57$

(b) $x_{n+1}=\frac{0.5}{1+x_n^4}, \quad x_0=0.48$

(c) $x_{n+1}=1+0.5\sin(x_n) \quad x_0=1.5$

14. 对缓慢收敛的序列, 艾特肯外推公式 (3.52) 能大大地加速其收敛. 使用下列算法:

$$\begin{aligned}
&x_1=g(x_0)\\
&x_2=g(x_1)\\
&x_3 \text{ 由} x_0,x_1\text{和}x_2\text{经艾特肯外推得到}\\
&x_4=g(x_3)\\
&x_5=g(x_4)\\
&x_6 \text{ 由} x_3,x_4\text{和}x_5\text{经艾特肯外推得到.}
\end{aligned}$$

继续这一过程. 将此算法应用于下列迭代:

(a) $x_{n+1}=2\mathrm{e}^{-x_n}, \quad x_0=0.8$

(b) $x_{n+1}=\frac{0.9}{1+x_n^4}, \quad x_0=0.75$

(c) $x_{n+1}=6.28+\sin(x_n), \quad x_0=6$

15. 证明 (3.52) 可以重写为

$$\alpha \approx x_n-\frac{(x_n-x_{n-1})^2}{(x_n-x_{n-1})-(x_{n-1}-x_{n-2})}$$

16. 对牛顿法 (3.58) 计算 (3.57). 试将你的计算结果与 3.2 节中 (3.19) 的结果作比较.

17. 当 $\alpha=g(\alpha)$、$g'(\alpha)=g''(\alpha)=0$ 和 $g^{(3)}(\alpha)\neq 0$ 时, 试问 (3.57) 应作何修正?

18. 当迭代

$$x_{n+1} = \frac{x_n(x_n^2+3a)}{3x_n^2+a}$$

收敛到不动点 $\alpha=\sqrt{a}$ 时, 收敛的阶是多少?

3.5 病态的求根问题

本节讨论 3.1 节到 3.4 节的方法不能很好地执行的两类问题. 通常这些问题的性质难以改善, 但是人们至少应该意识到问题的存在以及问题病态的原因.

我们从有**多重根**(multiple root) 的函数开始讨论. 对某个连续函数 $h(x)$, $h(\alpha)\neq 0$, m 是正整数, 如果

$$f(x)=(x-\alpha)^m h(x) \tag{3.59}$$

则 α 称为 $f(x)$ 的 m 重根. 如果假设 $f(x)$ 充分光滑, 则 m 重根的一种等价的定义是

$$f(\alpha)=f'(\alpha)=\cdots=f^{(m-1)}(\alpha)=0,\quad f^{(m)}(\alpha)\neq 0 \tag{3.60}$$

重数为 $m=1$ 的根称为**单根**(simple root)[回顾 3.2 节的 (3.18)].

例 3.5.1 (a) $f(x)=(x-1)^2(x+2)$ 有两个根. 根 $\alpha=1$ 的重数是 2, 根 $\alpha=-2$ 是单根.

(b) $f(x)=x^3-3x^2+3x-1$ 有 3 重根 $\alpha=1$. 为了看出这一点, 只要注意到

$$f(1)=f'(1)=f''(1)=0,\quad f'''(1)=6$$

根据 (3.60) 就可以得到结论.

(c) $f(x)=1-\cos(x)$ 的根 $\alpha=0$ 的重数 $m=2$. 为了看出这一点, 改写

$$f(x)=x^2\left[\frac{2\sin^2(x/2)}{x^2}\right]\equiv x^2h(x)$$

并且规定 $h(0)=\frac{1}{2}$. 函数 $h(x)$ 关于 x 连续. ■

当牛顿法和割线法用于计算重根 α 时, $\alpha-x_n$ 收敛到零的速度要比单根的情况慢很多. 另外, 由于 $f(x)$ 求值中的噪声, 在重根的附近会存在一个较大的**不定区间**(interval of uncertainty).

多重根附近的这个较大的不定区间, 对于数值求根而言是个严重的问题. 第 2 章的图 2-2 中, 举例说明了 $f(x)=(x-1)^3$ 求值的噪声, 注意到 $\alpha=1$ 是 $f(x)$ 的 3 重根. 图形也显示了 $\alpha=1$ 附近存在的较大的不定区间. 图 3-7 进一步解释单根和重根的不定区间之间的差别. $y=f(x)$ 图形两边的虚线表示 $f(x)$ 求值中噪声的外

侧极限, 两个图形中使用相同的噪声外侧极限 (这指的是对任意 x 的垂直距离, 而且它们是相同的). 噪声带和 x 轴的交集表明根 α 所在的区间, 而多重根所在的区间要大得多.

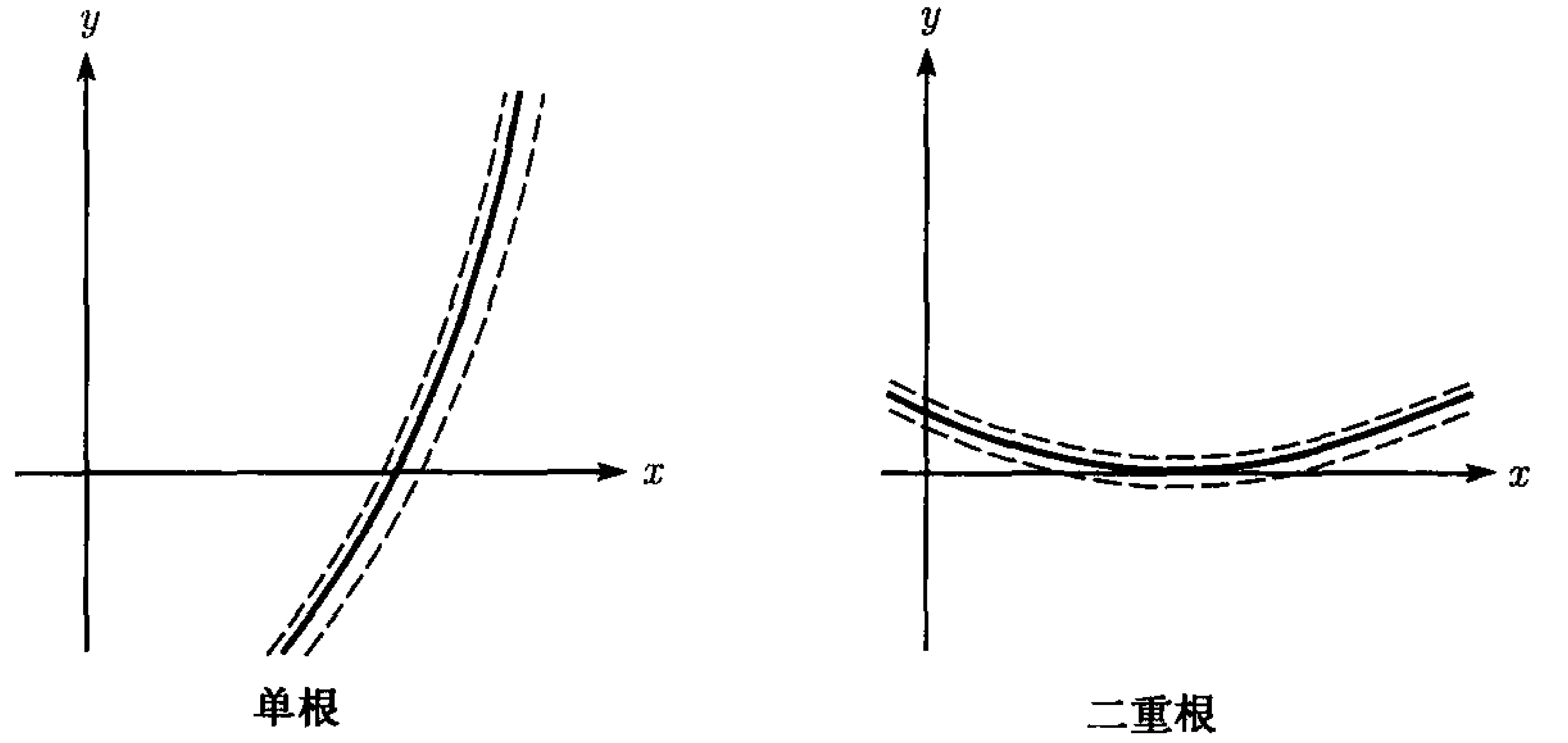


图 3-7 求根计算的不定区间

例 3.5.2 为了说明多重根对求根方法的影响, 我们用牛顿法求下列函数的根 $\alpha = 1.1$:

$$\begin{aligned} f(x) &= (x-1.1)^3(x-2.1) \\ &= 2.795\,1 + x(-8.954 + x(10.56 + x(-5.4 + x))) \end{aligned} \tag{3.61}$$

计算机采用保留 6 位有效数字的十进制, 并使用舍入. 函数 $f(x)$ 用 (3.61) 的嵌套形式求值, 类似地求 $f'(x)$ 值. 计算结果在表 3-7 中给出. 比率

$$\frac{\alpha - x_n}{\alpha - x_{n-1}} \tag{3.62}$$

的值在表中的 "比率" 列给出, 可以看出这些值大约等于 $\frac{2}{3}$. 迭代以 $\frac{2}{3}$ 的速率线性收敛. ■

表 3-7 (3.61) 的牛顿法

n	x_n	$f(x_n)$	$\alpha - x_n$	比率
0	0.800 000	0.035 10	0.300 000	
1	0.892 857	0.010 73	0.207 143	0.690
2	0.958 176	0.003 25	0.141 824	0.685
3	1.003 44	0.000 99	0.096 56	0.681
4	1.034 86	0.000 29	0.065 14	0.675
5	1.055 81	0.000 09	0.044 19	0.678
6	1.070 28	0.000 03	0.029 72	0.673
7	1.080 92	0.0	0.019 08	0.642

当用牛顿法计算 m 重根时, 可以证明, 比率 (3.62) 趋向于

$$\lambda=\frac{m-1}{m},\quad m\geqslant 1 \tag{3.63}$$

(证明留作习题 3.) 因此, 当 x_n 趋向于 α 时, 有

$$\alpha-x_n\approx\lambda(a-x_{n-1}) \tag{3.64}$$

而且误差大约以常数速率减少. 在前面的例子中, 因为根的重数是 3, 所以 $\lambda=\frac{2}{3}$, 它对应于表中最后一列的值. 误差公式 (3.64) 表明牛顿法的收敛速度比平常情况慢很多. 对于任何的 m 重根 $(m\geqslant 2)$, 我们有 $\lambda\geqslant\frac{1}{2}$; 因此, 对于多重根而言, 对分法至少和牛顿法一样快. 当然, m 必须是奇数以保证 $f(x)$ 在 $x=\alpha$ 处变号, 这样才可以使用对分法.

进一步观察表 3-7 可以看出, 虽然迭代 x_7 还离 $\alpha=1.1$ 很远, 但它已是计算机中 $f(x)$ 的精确根. 这可由 $f(x)$ 求值中出现的舍入误差来解释. $f(x)$ 求值产生的噪声导致根 α 可能位于其中的一个相对大的区间, 如前面图 3-7 所显示的那样.

获得重根精确值的唯一办法是用解析方法去掉多重性, 得到一个以 α 为单根的新函数. 否则, 在根所在之处有一个较大的不定区间. 为了去掉多重性, 我们可以先用牛顿法确定 α 的重数 m. 利用 (3.63) 和 (3.64), 再结合逼近

$$\lambda\approx\lambda_n\equiv\frac{x_n-x_{n-1}}{x_{n-1}-x_{n-2}} \tag{3.65}$$

(参见习题 3.) 一旦求出 λ, 就可以由 (3.63) 确定 m. 定义新函数

$$F(x)=f^{(m-1)}(x)$$

根据 (3.59), 可以证明 $F(x)$ 以 α 为单根. 通过解 $F(x)=0$ 可精确地求出 α.

例 3.5.3　对 (3.61) 两次求导得到新的求根问题

$$f''(x)\equiv 21.12-32.4x+12x^2=0$$

此方程以 $\alpha=1.1$ 为单根, 所以牛顿法能迅速地收敛到一个很精确的值. 作为解方程 $f''(x)=0$ 的初值, 我们可用表 3-7 中最后的计算值 x_7. ■

根的稳定性

对大多数函数 $f(x)$ 而言, 如果在计算函数时产生很小的误差, 则函数的根也会相应地改变一个小量. 然而, 也有一些函数的情况并非如此. 对那些函数来说, $f(x)$ 求值时的一个很小的误差会导致函数的根很大的变化. 这种函数的求根称为*病态问题*或*不稳定问题*. 我们先给出这种函数的一个众所周知的例子, 然后再分析不稳定求根问题.

例 3.5.4 定义

$$\begin{aligned}f(x)&=(x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7)\\&=x^7-28x^6+322x^5-1\ 960x^4+6\ 769x^3-13\ 132x^2+13\ 068x-5\ 040\end{aligned}\tag{3.66}$$

接着把 x^6 的系数从 -28 变为 -28.002, 得到一个新函数 $F(x)$. 系数的变化相对较小

$$|\mathrm{Rel}(28.002)|=\frac{0.002}{28}\doteq 7.14\times 10^{-5}\tag{3.67}$$

$f(x)$ 的根显然是 $\{1,2,\cdots,7\}$. 表 3-8 给出了 $F(x)$ 的舍入到 8 位有效数字的根. 虽然从 $f(x)$ 到 $F(x)$ 的变化相对较小, 但 $F(x)$ 的一些根与对应的 $f(x)$ 的根相去甚远. $f(x)$ 的其他一些系数类似的变化也将对根导致同样的结果. ■

表 3-8 $f(x)$ 和 $F(x)$ 的根

$f(x)$ 的根	$F(x)$ 的根	误差
1	1.000 002 8	−2.8E−6
2	1.998 938 2	1.1E−3
3	3.033 125 3	−3.3E−2
4	3.819 569 2	0.18
5	5.458 675 8+0.540 125 78i	−0.46−0.54i
6	5.458 675 8−0.540 125 78i	0.54+0.54i
7	7.233 012 8	−0.23

为了讨论近似计算函数 $f(x)$ 的值, 我们引入扰动函数

$$F_\varepsilon(x)=f(x)+\varepsilon g(x)\tag{3.68}$$

假设函数 $g(x)$ 连续可微, ε 是一个适当小的数. 对小的 ε 值, 函数 $F_\varepsilon(x)$ 和 $f(x)$ 几乎相同.

例 3.5.5 对于前面的例题 (3.66), 有

$$F_\varepsilon(x)=f(x)+\varepsilon g(x),\quad g(x)=x^6,\quad \varepsilon=-0.002 \qquad ■\tag{3.69}$$

$F_\varepsilon(x)$ 的根与 ε 有关, 用 $\alpha(\varepsilon)$ 表示这样的根. $f(x)$ 原来的根 α 恰好是 $\alpha(0)$. 为了简化讨论, 假设 $\alpha(0)$ 是 $f(x)$ 的一个单根, 从而 $f'(\alpha(0))\neq 0$. 这里的讨论可用于理解例 3.5.4 中出现的问题. 在上述假设下, 能够证明当 ε 充分小时, 可以用泰勒多项式逼近来估计 $\alpha(\varepsilon)$. 我们将利用

$$\alpha(\varepsilon)\approx\alpha(0)+\varepsilon\alpha'(0)\tag{3.70}$$

这样, 我们需要计算 $\alpha'(0)$.

由于 $\alpha(\varepsilon)$ 是 $F_\varepsilon(x)$ 的根, 所以对小的 ε, 有

$$f(\alpha(\varepsilon)) + \varepsilon g(\alpha(\varepsilon)) = 0 \tag{3.71}$$

对上述方程两边关于 ε 求导, 有

$$f'(\alpha(\varepsilon))\alpha'(\varepsilon) + g(\alpha(\varepsilon)) + \varepsilon g'(\alpha(\varepsilon))\alpha'(\varepsilon) = 0 \tag{3.72}$$

代入 $\varepsilon = 0$, 得到

$$f'(\alpha(0))\alpha'(0) + g(\alpha(0)) = 0$$

解出 $\alpha'(0)$

$$\alpha'(0) = -\frac{g(\alpha(0))}{f'(\alpha(0))} \tag{3.73}$$

把上式用到 (3.70) 中, 对所有充分小的 ε 值, 得到

$$\alpha(\varepsilon) \approx \alpha(0) - \varepsilon\frac{g(\alpha(0))}{f'(\alpha(0))} \tag{3.74}$$

如果导数 $\alpha'(0)$ 的值很大, 则稍稍改变 ε 会对根产生很大的影响.

例 3.5.6 考虑 (3.66) 中多项式 $f(x)$ 的根 $\alpha(0) = 4$. 根据 (3.69) 中 $F_\varepsilon(x)$ 的定义, 有

$$\begin{aligned} f'(4) &= (4-1)(4-2)(4-3)(4-5)(4-6)(4-7) = -36 \\ g(4) &= 4^6 = 4\,096 \\ \alpha'(0) &= \frac{4\,096}{-36} \doteq 114 \\ \alpha(\varepsilon) &\approx 4 + 114\varepsilon \end{aligned} \tag{3.75}$$

这表明稍稍改变 ε, 在 $f(x)$ 的根 $\alpha(0) = 4$ 所对应的根 $\alpha(\varepsilon)$ 中, ε 的影响被放大了很多倍. 对于 (3.69) 中具体的选择,

$$\alpha(\varepsilon) \doteq 4 + 114(-0.002) = 3.772$$

这是表 3-8 中给出的实际根 3.820 的一个近似. 对于表 3-8 中 $\alpha(0) = 5$ 或者 $\alpha(0) = 6$ 的结果, 预估公式 (3.74) 是无效的. 这是因为 (3.74) 只适用于实数扰动, 不能根据它预估表 3-8 中的复数扰动. 为了使 (3.74) 有效, ε 的值必须更小. ■

如同 (3.66) 那样的多项式求根问题是一个不稳定问题. 对于这样的问题除了采用更高精度的算术运算之外几乎无计可施. 主要的困难在于所求数学方程的原始表达式, 通常我们应当采用别的途径来解决原来的数学问题以便能防止不稳定的情况发生.

习题

1. 用牛顿法计算下列方程的根

$$f(x) = x^5 + 0.9x^4 - 1.62x^3 - 1.458x^2 + 0.656\ 1x + 0.590\ 49$$

打印迭代和函数的值. 用逼近 (3.65) 给出 (3.62) 的比值

$$\frac{\alpha - x_n}{\alpha - x_{n-1}} \approx \frac{x_{n+1} - x_n}{x_n - x_{n-1}}$$

选择几个不同的 x_0 重复本习题. 给出一些对于求根问题来说看起来重要的见解.

注: 和 (3.63) 中一样, 上述过程首先要逼近 $\lambda = (m-1)/m$. 但是当 x_n 离根较近时, 由于 $f(x)$ 求值中的噪声, 这些逼近将会偏离 λ.

2. 对下列多项式重复习题 1:

$$f(x) = x^4 - 3.2x^3 + 0.96x^2 + 4.608x - 3.456$$

3. 利用 3.4 节中的不动点迭代理论, 推导 (3.63) 到 (3.65) 的结果. 为此, 首先写出

$$x_{n+1} = g(x_n) \equiv x_n - \frac{f(x_n)}{f'(x_n)}$$

然后由 (3.59) 得

$$g(x) = x - \frac{(x-\alpha)h(x)}{h(x) + (x-\alpha)h'(x)}$$

再利用推论 3.4.3.

4. 继续例 3.5.6, 对根 $\alpha(0) = 3$ 和 $\alpha(0) = 7$ 的情形分别计算根 $\alpha(\varepsilon)$.

5. 对 (3.66) 中另一种变化做扰动计算. 把 x^4 的系数从 $-1\ 960$ 变为 $-1\ 960.14$. 系数的相对扰动是多少? 对 $\alpha(0) = 3$ 和 $\alpha(0) = 5$ 计算 $\alpha(\varepsilon)$.

6. 考虑求解 $x/(1+x) - 0.99 = 0$ 的问题, 称其根为 α. 设 $\alpha(\varepsilon)$ 是 $x/(1+x) - 0.99 + \varepsilon = 0$ 的解.

(a) 用 (3.74) 估计 $\alpha(\varepsilon) - \alpha$.

(b) 直接计算 $\alpha(\varepsilon)$, 再计算 $\alpha(\varepsilon) - \alpha$ 并且与 (a) 作比较. 评价你的结果.

7. 考虑多项式 $f(x) = x^5 - 300x^2 - 126x + 5\ 005$, 它有一个根 $\alpha = 5$. 再考虑扰动函数

$$F_\varepsilon(x) = f(x) + \varepsilon x^5 = (1+\varepsilon)x^5 - 300x^2 - 126x + 5\ 005$$

其中 ε 是一个很小的数. 设 $\alpha(\varepsilon)$ 表示对应于 $\alpha(0) = 5$ 的方程 $F_\varepsilon(x)=0$ 的扰动根. 估计 $\alpha(\varepsilon) - 5$. 求方程 $f(x) = 0$ 的根 $\alpha = 5$ 是否是一个稳定问题?

8. 使用牛顿法求某个方程 $f(x) = 0$ 的根. 下表给出的最初几步迭代, 显示收敛速度很慢. 基于这些数值结果, 关于根 α 你能说些什么呢? 若已知 $f(x)$, 你如何求出 α 的精确值?

n	x_n	$x_n - x_{n-1}$
0	0.75	
1	0.752 710	0.002 71
2	0.754 795	0.002 08
3	0.756 368	0.001 57
4	0.757 552	0.001 18
5	0.758 441	0.000 889

第 4 章　插值和逼近

数学课程中遇到的大多数函数都不能精确求值, 即使我们通常使用这些函数好像它们的值是已知的. 这些函数中最简单最重要的有 $\sqrt{x}$、e^x、$\ln(x)$ 和三角函数, 还有许多在物理学、工程学和其他学科中常见的函数.

无论是手算还是用计算机计算函数值, 我们基本上只用初等算术运算 $+$、$-$、$\times$ 和 $\div$. 利用这些运算, 我们就可以计算多项式函数和*有理函数*(rational function) 的值. 有理函数指多项式除以多项式. 更复杂函数的计算必须利用多项式或者有理函数包括它们的分段变形 (即样条函数) 来逼近. 本章我们讨论函数的多项式逼近. 一般来说有理函数可给出更有效的逼近, 但是对大多数问题多项式逼近是适当的, 它们的理论讨论也更容易, 因此我们只讨论多项式逼近.

插值过程就是找一个函数使它的图像经过一组给定的点, 并计算这个函数值. 这些给定的点可能取自一个物理问题的测量值, 或者由一个已知函数得到. 插值函数通常选用限定的几类函数, 多项式是其中最常用的一类. 我们在 4.1 节定义多项式插值问题, 并给出构造插值多项式的两种公式. 4.2 节分析多项式插值中的误差. 在过去的几十年中, 分段多项式函数得到广泛的应用, 其中最主要的一类函数就是*样条函数*(spline function). 4.3 节引入插值中的样条函数.

在第 1 章, 我们学习了用泰勒多项式作为求给定函数 $f(x)$ 的值的工具. 对任意要求的精度, 逼近大多数常用函数, 用泰勒多项式是一个相对容易的方法. 然而, 对 $f(x)$ 来说, 泰勒多项式通常是一种效能很低的逼近; 如果这个逼近要使用很多次, 那么应该用一个需要较少计算时间的公式来代替它. 相对于一个泰勒多项式, 通常存在另一个具有同样精度但次数低很多的多项式, 较低的次数可以减少计算次数, 降低求值过程中的舍入误差.

在本章的最后四节中, 我们考虑用多项式逼近函数 $f(x)$ 的一般问题. 这将扩展第 1 章所讨论的泰勒多项式逼近的思想. 4.4 节中讨论用多项式逼近一个函数的一般问题, 4.6 中我们用插值构造一个改进的泰勒逼近. 4.5 节中介绍一类重要的多项式, 即*切比雪夫多项式*(Chebyshev polynomial), 这些多项式在 4.6 节的构造过程中会被用到. 最后在 4.7 节介绍了*最小二乘*(least square) 逼近的思想.

4.1　多项式插值

插值有多种用途. 最初, 它被广泛地用于常用的数学函数表中, 但随着计算机

和计算器的普及, 目前这种用途已经微不足道了. 但是在扩展仅仅定义在离散点集上函数的相关问题中仍然使用插值, 这些问题通常出现在数值求解微分方程和积分方程中. 其次, 应用插值可解决逼近理论中的一些问题. 插值也是构造常用函数的可计算逼近的重要工具. 此外, 为了数值积分或者数值微分一个函数, 常用一个具有较简单形式的逼近表达式代替这个函数, 然后对其进行积分或者微分. 这些较简单的表达式几乎都通过插值来得到. 并且, 一些广泛使用的求解微分方程的数值方法也是从插值逼近得到的. 最后, 插值被广泛地用于计算机图形处理, 当只有一些关于被研究的几何对象的数据点时, 应用插值可产生光滑的曲线或曲面.

4.1.1 线性插值

首先考虑构造一个多项式, 它的图像要经过两个给定的数据点. 这种插值形式称为**线性插值**(linear interpolation), 讨论它也为以后研究更一般的多项式插值做准备.

给定两个点 (x_0, y_0) 和 (x_1, y_1) 并且 $x_0 \neq x_1$, 画一条直线通过这两点, 如图 4-1 所示. 这条直线是下列线性多项式的图像

$$P_1(x) = \frac{(x_1 - x)y_0 + (x - x_0)y_1}{x_1 - x_0} \tag{4.1}$$

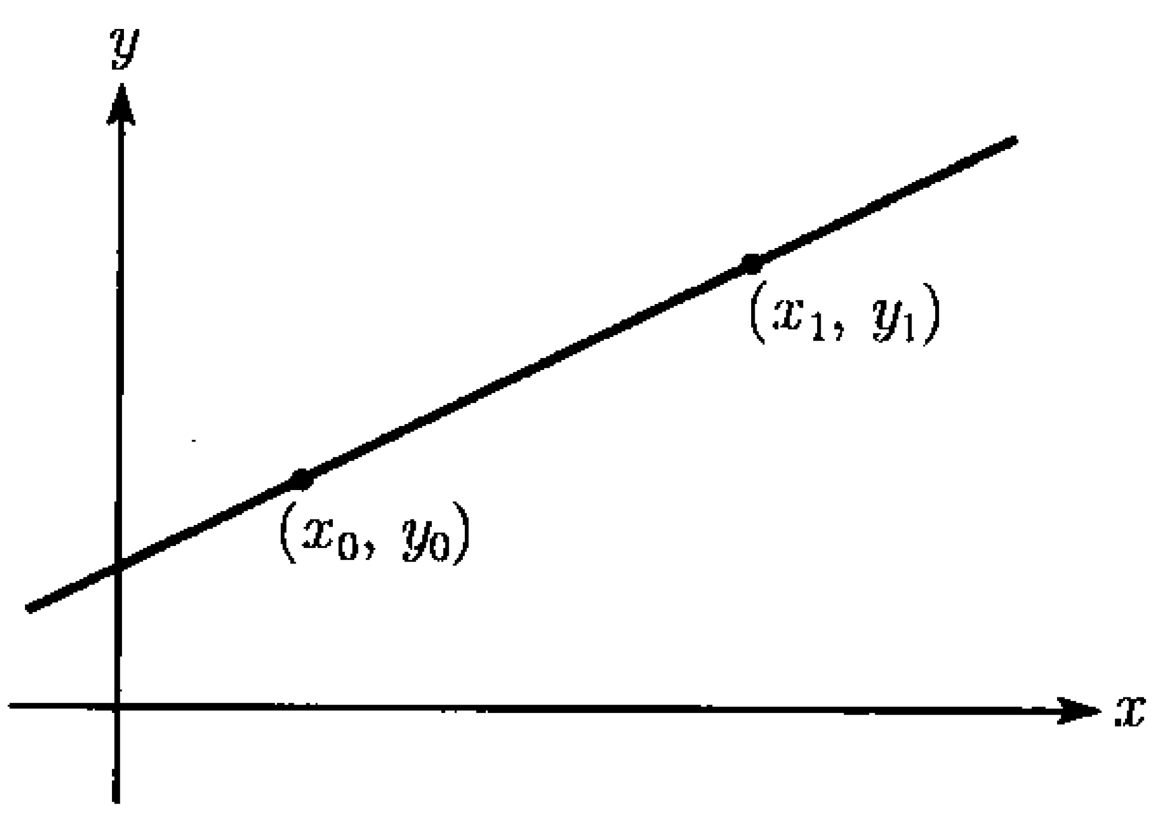


图 4-1 线性插值

读者应该验证这个函数的图像是由 (x_0, y_0) 和 (x_1, y_1) 确定的直线. 我们说该函数在点 x_i 处插值数值 y_i, $i = 0, 1$; 即

$$P_1(x_i) = y_i, \quad i = 0, 1$$

例 4.1.1 设数据点是 $(1, 1)$ 和 $(4, 2)$. 多项式 $P_1(x)$ 为

$$P_1(x) = \frac{(4 - x)(1) + (x - 1)(2)}{3} \tag{4.2}$$

$y = P_1(x)$ 和 $y = \sqrt{x}$ 的图像如图 4-2 所示, 数据点从函数 $y = \sqrt{x}$ 得到. ■

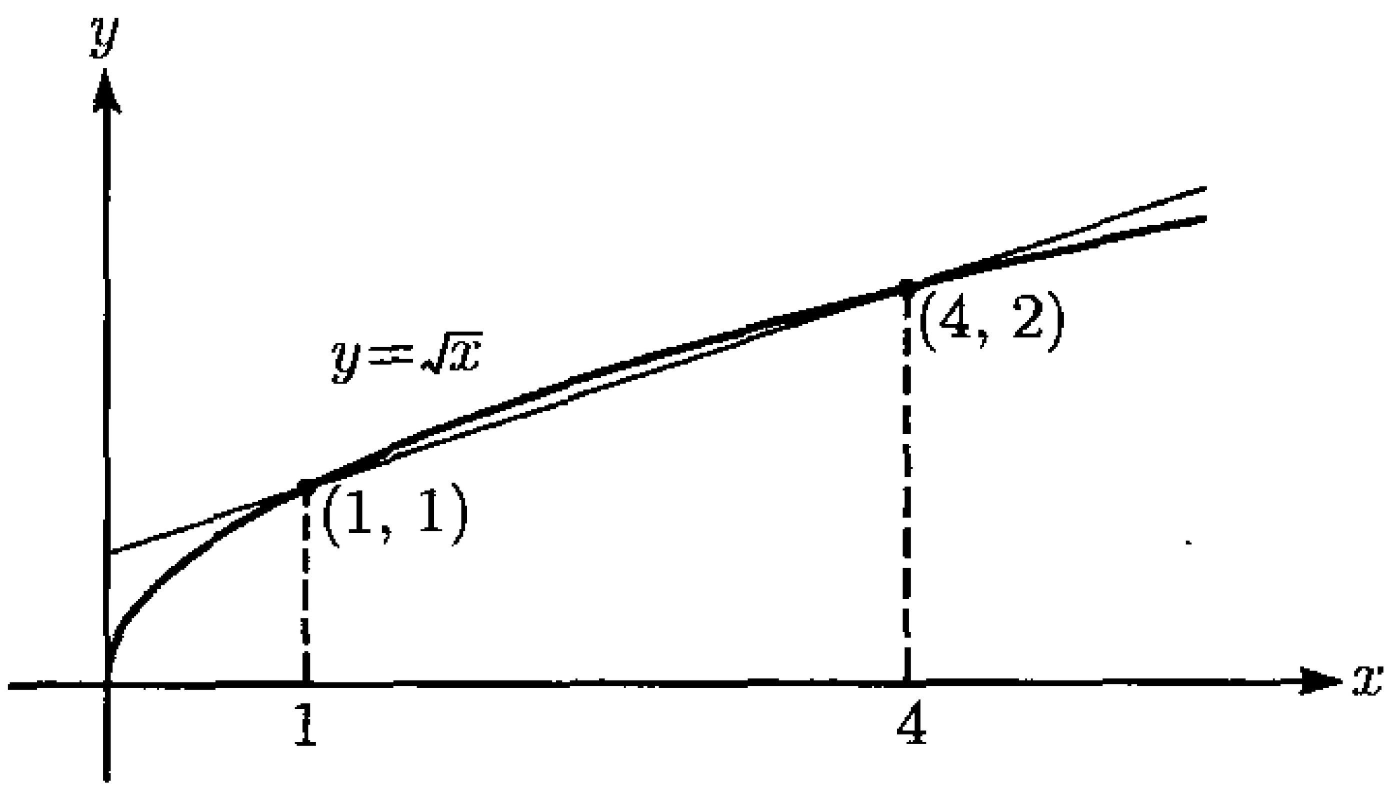


图 4-2 $y=\sqrt{x}$ 和它的插值多项式 (4.2)

例 4.1.2 我们来利用函数值

$$e^{0.82} \doteq 2.270\ 500, \quad e^{0.83} \doteq 2.293\ 319$$

得到 $e^{0.826}$ 的一个估计. 记 $x_0=0.82$ 及 $x_1=0.83$. 在点 x_0 和 x_1 处插值 e^x 的多项式 $P_1(x)$ 是

$$P_1(x)=\frac{(0.83-x)(2.270\ 500)+(x-0.82)(2.293\ 319)}{0.01} \tag{4.3}$$

特别地,

$$P_1(0.826)=2.284\ 191\ 4 \tag{4.4}$$

为了做比较, 我们给出具有 8 位有效数字的真值:

$$e^{0.826} \doteq 2.284\ 163\ 8 \quad\blacksquare \tag{4.5}$$

4.1.2 二次插值

大多数数据不是来自直线, 而是出现在曲线的图像中. 为了更好地逼近这种情况, 我们考虑次数大于 1 的多项式. 假设给定三个数据点 (x_0,y_0), (x_1,y_1) 和 (x_2,y_2), 并且 x_0, x_1 和 x_2 是不同的数. 我们构造如下经过这些点的二次多项式:

$$P_2(x)=y_0L_0(x)+y_1L_1(x)+y_2L_2(x) \tag{4.6}$$

其中

$$\begin{aligned}L_0(x)&=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}\\ L_1(x)&=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}\\ L_2(x)&=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}\end{aligned} \tag{4.7}$$

公式 (4.6) 称为二次插值多项式的**拉格朗日公式**(Lagrange's formula); 多项式 $L_0(x)$, $L_1(x)$ 和 $L_2(x)$ 称为**拉格朗日插值基函数**(Lagrange interpolation basis functions).

对于 $i=0,1,2$, 每个多项式 $L_i(x)$ 都是二次的, 从而 $P_2(x)$ 的次数 $\leqslant 2$. 另外, 对 $0\leqslant i,j\leqslant 2$, 有

$$L_i(x_j)=0,\quad j\neq i$$
$$L_i(x_i)=1$$

合并上述两式得到

$$L_i(x_j)=\delta_{ij},\quad 0\leqslant i,j\leqslant 2 \tag{4.8}$$

其中 δ_{ij} **是克罗内克 δ 函数**(Kronecker delta function)

$$\delta_{ij}=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases}$$

利用这些性质, 容易证明 $P_2(x)$ 插值数据

$$P_2(x_i)=y_i,\quad i=0,1,2$$

例 4.1.3 对数据点 $(0,-1)$, $(1,-1)$ 和 $(2,7)$ 构造 $P_2(x)$. 则有

$$P_2(x)=\frac{(x-1)(x-2)}{2}(-1)+\frac{x(x-2)}{-1}(-1)+\frac{x(x-1)}{2}(7) \tag{4.9}$$

图 4-3 给出其图像. ■

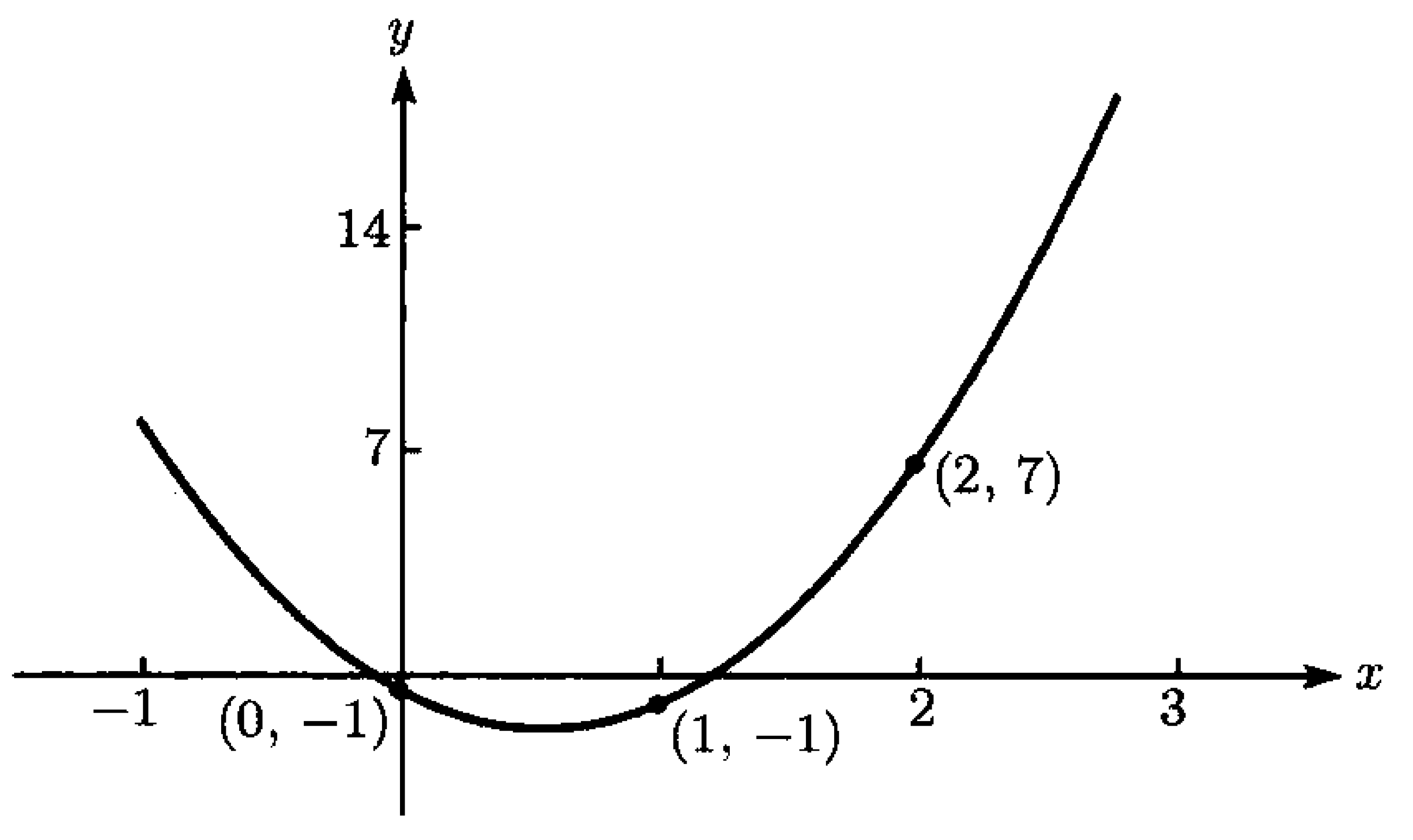


图 4-3 二次插值多项式 (4.9)

就线性插值来说, 显然仅有一条直线经过两个给定的数据点. 而给定三个数据点时, 是否仅有一个二次插值多项式其图像经过这些点却不够明显. 为了说明仅有一个这样的多项式, 假设存在另一个这样的多项式, 然后证明它一定等于 $P_2(x)$. 设 $Q_2(x)$ 表示另一个次数 $\leqslant 2$ 的多项式且它的图像经过 (x_i,y_i), 其中 $i=0,1,2$. 定义

$$R(x)=P_2(x)-Q_2(x) \tag{4.10}$$

因为 P_2 和 Q_2 的次数 $\leqslant 2$, 所以也有 $\deg(R) \leqslant 2$. 另外, 利用 P_2 和 Q_2 的插值性质得到

$$\begin{aligned} R(x_i) &= P_2(x_i) - Q_2(x_i) \\ &= y_i - y_i = 0 \end{aligned}$$

因此, $R(x)$ 有三个不同的根 x_0, x_1 和 x_2, 而它的次数 $\leqslant 2$. 根据代数基本定理 [见附录 B 中 B.1(e)], 这是不可能的, 除非 $R(x) \equiv 0$, 即 $R(x)$ 是零多项式. 由 (4.10) 知 $P_2(x) \equiv Q_2(x)$, 因此仅有一个次数 $\leqslant 2$ 的多项式满足

$$P_2(x_i) = y_i, \quad i = 0, 1, 2 \tag{4.11}$$

例 4.1.4 根据函数值

$$\mathrm{e}^{0.82} \doteq 2.270\,500, \quad \mathrm{e}^{0.83} \doteq 2.293\,319, \quad \mathrm{e}^{0.84} \doteq 2.316\,367$$

的二次插值来计算 $\mathrm{e}^{0.826}$. 取 $x_0 = 0.82$, $x_1 = 0.83$ 及 $x_2 = 0.84$. 则可以直接计算得到 8 位有效数字的近似值

$$P_2(0.826) \doteq 2.284\,163\,9 \tag{4.12}$$

与真解 $\mathrm{e}^{0.826} \doteq 2.284\ 163\ 8$ 相比, $P_2(0.826)$ 大大地改进了近似值 $P_1(0.826) \doteq 2.284\ 191\ 4$. ■

4.1.3 高次插值

现在考虑一般情况. 假设给定 $n+1$ 个数据点 $(x_0, y_0), \cdots, (x_n, y_n)$, 并且所有 x_i 互不相同. 给出次数 $\leqslant n$ 的插值多项式如下

$$P_n(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x) \tag{4.13}$$

对于 $i = 0, 1, \cdots, n$, 每个 n 次多项式 $L_i(x)$ 为

$$L_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \tag{4.14}$$

其分母等于当 $x = x_i$ 时分子的值. 对于 $i = 0$ 的情形, 公式为

$$L_0(x) = \frac{(x - x_1) \cdots (x - x_n)}{(x_0 - x_1) \cdots (x_0 - x_n)}$$

类似地可以定义 $L_n(x)$. 注意到 (4.14) 中定义的函数 $L_i(x)$ 与 n 有关. 由于 n 的值通常可从上下文得到, 所以我们不用显式表达式表示 $L_i(x)$ 对 n 的依赖关系.

对于 $i=0,1,\cdots,n$, 根据 (4.14), 容易证明 $L_i(x)$ 满足

$$L_i(x_j)=\delta_{ij}, \quad 0\leqslant j\leqslant n \tag{4.15}$$

如果利用 (4.15), 把 $x=x_j$ 直接代入 (4.13) 可以得到

$$P_n(x_j)=y_j, \quad j=0,1,\cdots,n \tag{4.16}$$

公式 (4.13) 称为 n 次插值多项式的**拉格朗日公式**(Lagrange's formula).

类似于二次插值, 可以证明所有次数 $\leqslant n$ 的多项式中仅有一个多项式满足 (4.16). 这个问题留作习题 11. 因为其重要性, 我们把本小节中的结果叙述为一个定理.

定理 4.1.5 设 $n\geqslant 0$, $x_0,x_1,\cdots,x_n$ 是 $n+1$ 个不同的数. 另外, 任意给定 $n+1$ 个数 $y_0,\cdots,y_n$. 则在所有次数 $\leqslant n$ 的多项式中有且仅有一个多项式 $P_n(x)$ 满足

$$P_n(x_i)=y_i, \quad i=0,1,\cdots,n$$

多项式插值在本书中的主要用途是对已知函数 $f(x)$ 给出逼近. 多项式很容易求值, 并且也可以直接对其进行积分和微分. 在 4.6 节和第 5 章中将利用插值多项式得出任意函数 $f(x)$ 的求值、积分和微分的近似方法.

插值的另一个重要用途是找到一条经过数据点集合 $\{(x_i,y_i)|i=1,\cdots,N\}$ 的**光滑曲线**. 一般来说, 我们希望曲线没有任何振荡和波动. 这个问题的求解就产生了将于 4.3 节介绍的样条函数. 多项式插值的内容在讨论这种新型插值函数的时候仍然是很有用的.

我们提醒读者, 插值多项式 $P_n(x)$ 的次数可能会小于 n. 为了说明这一点, 对于 $i=0,1,2$, 假设三个数据点 (x_i,y_i) 在一条直线上. 那么 $P_2(x)\equiv P_1(x)$, 其中 $P_1(x)$ 是由点 (x_0,y_0) 和 (x_1,y_1) 确定的线性插值多项式. 结论成立的原因在于 (4.10) 式后面证明的唯一性. 根据假设条件, 多项式 $P_1(x)$ 在三个数据点上插值, 因而由 $P_2(x)$ 的唯一性, 它一定等于 $P_2(x)$. 对于高阶多项式插值也有类似的结果.

4.1.4 差商

拉格朗日公式 (4.13) 很适宜于讨论插值理论方面的很多问题, 但在实际计算插值多项式的值时, 它却不太令人满意. 例如, 知道了 $P_2(x)$ 并不能帮助计算 $P_3(x)$. 正因为这种原因, 我们将介绍另一种更容易计算插值多项式 $P_1(x),P_2(x),\cdots,P_n(x)$ 的公式.

作为这个新的插值公式的准备, 我们引入函数 $f(x)$ 导数的离散形式. 设 x_0 和 x_1 是不同的数, 定义 $f(x)$ 的**一阶差商**(first-order divided difference) 为

$$f[x_0,x_1]=\frac{f(x_1)-f(x_0)}{x_1-x_0} \tag{4.17}$$

如果 $f(x)$ 在包含 x_0 和 x_1 的区间上可微, 则由中值定理可以推出

$$f[x_0, x_1] = f'(c) \tag{4.18}$$

其中 c 在 x_0 和 x_1 之间. 这说明用 $f[x_0, x_1]$ 模拟 $f(x)$ 的导数是合理的, 而且当 x_0 和 x_1 很靠近时, 有

$$f[x_0, x_1] \approx f'\left(\frac{x_0 + x_1}{2}\right) \tag{4.19}$$

一般来说这是一个很精确的逼近. 习题 20 中将给出这个逼近的分析过程.

例 4.1.6 设 $f(x) = \cos(x)$, $x_0 = 0.2$, $x_1 = 0.3$. 则

$$f[x_0, x_1] = \frac{\cos(0.3) - \cos(0.2)}{0.3 - 0.2} = -0.247\ 300\ 9 \tag{4.20}$$

从 (4.19) 可得

$$f'\left(\frac{x_0 + x_1}{2}\right) = -\sin(0.25) \doteq -0.247\ 404\ 0$$

$f[x_0, x_1]$ 是这个导数的一个很好的逼近. ■

我们用低阶差商递归地定义高阶差商. 设 x_0, x_1, x_2 是互不相同的实数, 定义

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \tag{4.21}$$

称它为*二阶差商*(second-order divided difference). 对于互不相同的 x_0, x_1, x_2, x_3, 定义

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} \tag{4.22}$$

这是*三阶差商*(third-order divided difference). 一般地, 设 $x_0, x_1, \cdots, x_n$ 是 $n+1$ 个互不相同的数, 定义

$$f[x_0, \cdots, x_n] = \frac{f[x_1, \cdots, x_n] - f[x_0, \cdots, x_{n-1}]}{x_n - x_0} \tag{4.23}$$

这是 n 阶差商, 有时也称它是*牛顿差商*(Newton divided difference).

下面的定理给出了差商和同阶导数之间的关系, 它的证明在下一节中给出.

定理 4.1.7 设 $n \geqslant 1$, 并且假设 $f(x)$ 在某个区间 $\alpha \leqslant x \leqslant \beta$ 上 n 次连续可微, $x_0, x_1, \cdots, x_n$ 是区间 $[\alpha, \beta]$ 内 $n+1$ 个互不相同的数. 则对位于 $x_0, x_1, \cdots, x_n$ 的最小值和最大值之间的某一个点 c, 有

$$f[x_0, x_1, \cdots, x_n] = \frac{1}{n!} f^{(n)}(c) \tag{4.24}$$

例 4.1.8 设 $f(x)=\cos(x)$, $x_0=0.2$, $x_1=0.3$, $x_2=0.4$. 则对 (4.20) 中给出的 $f[x_0,x_1]$, 有

$$f[x_1,x_2]=\frac{\cos(0.4)-\cos(0.3)}{0.4-0.3}\doteq -0.342\ 755\ 0$$

由 (4.21) 知

$$f[x_0,x_1,x_2]\doteq\frac{-0.342\ 755\ 0-(-0.247\ 300\ 9)}{0.4-0.2}=-0.477\ 270\ 5 \tag{4.25}$$

对 $n=2$ 的情形, (4.24) 变为

$$f[x_0,x_1,x_2]=\frac{1}{2}f''(c) \tag{4.26}$$

其中 c 在 x_0, x_1, x_2 的最小值和最大值之间. 用 $x=x_1$ 计算 $f''(x)=-\cos(x)$, 有

$$\frac{1}{2}f''(0.3)=-\frac{1}{2}\cos(0.3)\doteq -0.477\ 668\ 2$$

这个值与 (4.25) 中的结果几乎相等. ■

4.1.5 差商的性质

差商 (4.23) 有许多特殊的性质, 利用这些性质可以简化很多工作. 首先, 设 $(i_0,i_1,\cdots,i_n)$ 表示整数 $(0,1,\cdots,n)$ 的一个排列 (或重排). 则可以证明

$$f[x_{i_0},x_{i_1},\cdots,x_{i_n}]=f[x_0,x_1,\cdots,x_n] \tag{4.27}$$

原始定义 (4.23), 似乎意味着 $f[x_0,\cdots,x_n]$ 的值依赖于 $x_0,x_1,\cdots,x_n$ 的次序, 但 (4.27) 表明这是不对的. (4.27) 的证明比较繁琐, 这里仅考虑 $n=1$ 和 $n=2$ 的情形. 对于 $n=1$, 有

$$f[x_1,x_0]=\frac{f(x_0)-f(x_1)}{x_0-x_1}=\frac{f(x_1)-f(x_0)}{x_1-x_0}=f[x_0,x_1]$$

对于 $n=2$, 可以展开 (4.21), 得到

$$\begin{aligned}f[x_0,x_1,x_2]=&\frac{f(x_0)}{(x_0-x_1)(x_0-x_2)}\\&+\frac{f(x_1)}{(x_1-x_0)(x_1-x_2)}+\frac{f(x_2)}{(x_2-x_0)(x_2-x_1)}\end{aligned} \tag{4.28}$$

如果交换 x_0, x_1, x_2 的值, 则上式右端的分式交换次序, 但其和保持不变. 读者可以通过交换 (4.27) 的右端项中的 x_0 和 x_1 来尝试一下. 我们把 (4.28) 的推导过程留作习题 22.

倘若 $f(x)$ 充分可微, 定义 (4.17)、定义 (4.21) 和定义 (4.23) 可推广到部分或全部结点相同的情形, 这是差商的第二个有用的性质. 例如, 定义

$$f[x_0,x_0]=\lim_{x_1\to x_0} f[x_0,x_1]=\lim_{x_1\to x_0}\frac{f(x_1)-f(x_0)}{x_1-x_0}$$
$$f[x_0,x_0]=f'(x_0)$$

对任意的 $n\geqslant 1$, 设 (4.24) 中所有结点都趋向于 x_0. 这导出定义

$$f[x_0,x_0,\cdots,x_0]=\frac{1}{n!}f^{(n)}(x_0) \tag{4.29}$$

其中左端项表示 n 阶差商, 它的所有结点都是 x_0.

对于只有一部分结点相同的情形, 我们可以利用定义 (4.27)、定义 (4.29) 和定义 (4.23) 推广差商的定义. 例如,

$$\begin{aligned}f[x_0,x_1,x_0]&=f[x_0,x_0,x_1]\\&=\frac{f[x_0,x_1]-f[x_0,x_0]}{x_1-x_0}=\frac{f[x_0,x_1]-f'(x_0)}{x_1-x_0}\end{aligned} \tag{4.30}$$

差商更多的性质留在本节的习题中探究.

MATLAB 程序: 计算差商 给定一组数值 $f(x_0),\cdots,f(x_n)$, 通常我们需要计算一组差商

$$f[x_0,x_1],\ f[x_0,x_1,x_2],\cdots,\ f[x_0,x_1,\cdots,x_n]$$

下面给出的 MATLAB 函数可用于这个目的, 这个函数称为

$$\texttt{divdif_y}=\texttt{divdif(x_nodes,y_values)}$$

注意到 MATLAB 不容许零作下标, 当给定结点 $\{x_0,x_1,\cdots,x_n\}$ 和对应的函数值后, 我们定义输入向量 x_nodes 和 y_nodes 如下:

$$\begin{aligned}x_\text{nodes}&=[x_0,x_1,\cdots,x_n]\\x_\text{nodes}(i)&=x_{i-1},\quad i=1,\cdots,n+1\\y_\text{values}&=[f(x_0),f(x_1),\cdots,f(x_n)]\\y_\text{values}(i)&=f(x_{i-1}),\quad i=1,\cdots,n+1\end{aligned} \tag{4.31}$$

```
function divdif_f = divdif(t_nodes,f_values)
%
% This is a function,
```

```
%        divdif_f = divdif(t_nodes,f_values)
% It calculates the divided differences of the function
% values given in the vector f_values, which are the values of
% some function f(t) at the nodes given in t_nodes.  On exit,
%        divdif_f(i) = f[t_1,...,t_i], i=1,...,m
% with m the length of t_nodes.  The input values t_nodes and
% f_values are not changed by this program.

divdif_f = f_values;
m = length(t_nodes);
for i=2:m
  for j=m:-1:i
    divdif_f(j) = (divdif_f(j)-divdif_f(j-1)) ...
                              /(t_nodes(j)-t_nodes(j-i+1));
  end
end
```

表 4-1 给出使用这个程序的一个例子.

表 4-1　$\cos(x)$ 的值和差商

i	x_i	$\cos(x_i)$	D_i
0	0.0	1.000 000	0.100 000 0E + 1
1	0.2	0.980 067	−0.996 671 1E − 1
2	0.4	0.921 061	−0.488 402 0E + 0
3	0.6	0.825 336	0.490 076 3E − 1
4	0.8	0.696 707	0.381 224 6E − 1
5	1.0	0.540 302	−0.396 204 7E − 2
6	1.2	0.362 358	−0.113 489 0E − 2

4.1.6　牛顿差商插值公式

从 4.1.4 节开始部分的讨论中不难看出, 对于实际计算次数不断增加的插值多项式序列来说, 拉格朗日公式 (4.13) 是很不方便的. 此外, 在对不同的 n 计算多项式 $P_n(x)$ 的过程中, 对一个特定的 n, $P_n(x)$ 的计算结果很少能被用来计算更高次的插值多项式. 利用 $P_n(x)$ 的另一个公式可以避免这些问题, 这个公式使用了被插值数据的差商.

对于 $i=0,1,\cdots,n$, 设 $P_n(x)$ 表示 $f(x_i)$ 在 x_i 处的插值多项式. 因而, $\deg(P_n)\leqslant n$ 并且

$$P_n(x_i)=f(x_i),\quad i=0,1,\cdots,n \tag{4.32}$$

于是, 插值多项式可写为

$$P_1(x)=f(x_0)+(x-x_0)f[x_0,x_1] \tag{4.33}$$

$$P_2(x)=f(x_0)+(x-x_0)f[x_0,x_1]$$

$$+ (x - x_0)(x - x_1)f[x_0, x_1, x_2] \tag{4.34}$$

$$\vdots$$

$$\begin{aligned} P_n(x) = & f(x_0) + (x - x_0)f[x_0, x_1] + \cdots \\ & + (x - x_0)(x - x_1)\cdots(x - x_{n-1})f[x_0, x_1, \cdots, x_n] \end{aligned} \tag{4.35}$$

这些公式称为插值多项式的**牛顿差商公式**(Newton's divided difference formula). 注意, 对 $k \geqslant 0$,

$$P_{k+1}(x) = P_k(x) + (x - x_0)\cdots(x - x_k)f[x_0, x_1, \cdots, x_k, x_{k+1}] \tag{4.36}$$

从而, 一旦计算出差商的系数, 就可以用最少的计算从 k 次插值多项式得到 $k+1$ 次插值多项式.

我们只考虑 (4.33) 和 (4.34) 的证明. 首先, 考虑 $P_1(x_0)$ 和 $P_1(x_1)$. 易得 $P_1(x_0) = f(x_0)$,

$$\begin{aligned} P_1(x_1) &= f(x_0) + (x_1 - x_0)\left[\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right] \\ &= f(x_0) + [f(x_1) - f(x_0)] = f(x_1) \end{aligned}$$

因此, $\deg(P_1) \leqslant 1$, 而且它满足插值条件. 根据插值多项式的唯一性 (参见定理 4.1.5), 公式 (4.33) 是 $f(x)$ 在点 x_0 和点 x_1 处的线性插值多项式.

对 (4.34), 注意到

$$P_2(x) = P_1(x) + (x - x_0)(x - x_1)f[x_0, x_1, x_2] \tag{4.37}$$

它满足 $\deg(P_2) \leqslant 2$, 而且对于 x_0 和 x_1, 有

$$P_2(x_i) = P_1(x_i) + 0 = f(x_i), \quad i = 0, 1$$

同样,

$$\begin{aligned} P_2(x_2) &= f(x_0) + (x_2 - x_0)f[x_0, x_1] + (x_2 - x_0)(x_2 - x_1)f[x_0, x_1, x_2] \\ &= f(x_0) + (x_2 - x_0)f[x_0, x_1] + (x_2 - x_1)\{f[x_1, x_2] - f[x_0, x_1]\} \\ &= f(x_0) + (x_1 - x_0)f[x_0, x_1] + (x_2 - x_1)f[x_1, x_2] \\ &= f(x_0) + \{f(x_1) - f(x_0)\} + \{f(x_2) - f(x_1)\} \\ &= f(x_2) \end{aligned}$$

根据插值多项式的唯一性, 这是 $f(x)$ 在 $\{x_0, x_1, x_2\}$ 的二次插值多项式.

一般公式 (4.35) 的证明远比我们这里要考虑的更为复杂. 关于公式 (4.35) 的详细推导以及差商更多的性质, 参看 Atkinson(1989, 第 3 章).

例 4.1.9　设 $f(x)=\cos(x)$. 表 4-1 包含一组结点 x_i、函数值 $f(x_i)$ 和差商

$$D_i = f[x_0,\cdots,x_i],\quad i\geqslant 0$$

利用前面给出的 MATLAB 程序`divdif`可以计算它们. 对于不同的 n 值, 表 4-2 包含 $P_n(x)$ 在点 $x=0.1, x=0.3, x=0.5$ 处的值. 表中的最后一行给出了 $f(x)$ 的真值. 有关的计算利用公式 (4.35) 通过下面给出的 MATLAB 程序`interp`来实现. ■

上面的例题用到了等距节点 x_i. 但是在一般情况下, 使用差商插值公式 (4.35) 不需要插值节点是等距的, 也不需要把它们排成特定的次序. 在 4.6 节中, 我们将对非等距节点使用公式 (4.35).

表 4-2　插值 cos(x)

	$P_n(0.1)$	$P_n(0.3)$	$P_n(0.5)$
1	0.990 033 3	0.970 099 9	0.950 166 4
2	0.994 917 3	0.955 447 8	0.876 906 1
3	0.995 064 3	0.955 300 8	0.877 641 3
4	0.995 007 1	0.955 335 1	0.877 584 1
5	0.995 003 0	0.955 336 9	0.877 582 3
6	0.995 004 1	0.955 336 5	0.877 582 5
True	0.995 004 2	0.955 336 5	0.877 582 6

为了用一种有效的方法计算 (4.35), 可以利用 1.3 节中嵌套乘法的一种变形算法. 首先, 把 (4.35) 写为

$$\begin{aligned}P_n(x)=&D_0+(x-x_0)D_1+(x-x_0)(x-x_1)D_2\\&+\cdots+(x-x_0)\cdots(x-x_{n-1})D_n\end{aligned}\tag{4.38}$$

其中

$$D_0=f(x_0),\quad D_i=f[x_0,\cdots,x_i],\quad i\geqslant 1$$

其次, 把 (4.38) 改写为嵌套形式

$$\begin{aligned}P_n(x)=&D_0+(x-x_0)[D_1+(x-x_1)[D_2+\cdots\\&+(x-x_{n-2})[D_{n-1}+(x-x_{n-1})D_n]\cdots]\end{aligned}\tag{4.39}$$

例如,

$$P_3(x)=D_0+(x-x_0)[D_1+(x-x_1)[D_2+(x-x_2)D_3]]$$

利用 (4.39), 仅用 n 次乘法就可以算出 $P_n(x)$. 对于一个固定次数 n 来说, 使用嵌套形式 (4.39) 更加方便一些. 若为了计算一系列不断增加次数的插值多项式, 还是用原始的公式 (4.35) 更为有效.

MATLAB 程序：计算牛顿差商形式的插值多项式 为了帮助构造如表 4-1 和表 4-2 中那样的例子, 我们给出了下列插值程序. 程序使用了插值多项式的牛顿差商的嵌套形式. 程序的前面调用本节中给出的程序`divdif`. 如同前面的`divdif`一样, MATLAB 不允许 0 作为下标, 参见 (4.31).

```
function p_eval = interp(x_nodes,divdif_y,x_eval)
%
% This is a function
%       p_eval = interp(x_nodes,divdif_y,x_eval)
% It calculates the Newton divided difference form of
% the interpolation polynomial of degree m-1, where the
% nodes are given in x_nodes, m is the length of x_nodes,
% and the divided differences are given in divdif_y.  The
% points at which the interpolation is to be carried out
% are given in x_eval; and on exit, p_eval contains the
% corresponding values of the interpolation polynomial.

m = length(x_nodes);
p_eval = divdif_y(m)*ones(size(x_eval));
for i=m-1:-1:1
  p_eval = divdif_y(i) + (x_eval - x_nodes(i)).*p_eval;
end
```

习题

1. 给定数据点 (0,2) 和 (1,1).

(a) 找出插值这些数据的直线.

(b) 试求插值这些数据的函数 $f(x)=a+be^x$.

提示：求出 a 和 b, 使得 $f(0)=2$, $f(1)=1$.

(c) 试求插值这些数据的函数 $f(x)=a/(b+x)$.

画出每一种情况的插值函数的图像.

2. (a) 求函数 $P(x)=a+b\cos(\pi x)+c\sin(\pi x)$, 使其插值数据

x	0	0.5	1
y	2	5	4

(b) 求插值这些数据的二次多项式.

画出每一种情况的插值函数的图像.

3. 求解某个微分方程时得到下表. 利用相邻结点 x_i 之间的线性插值, 给出这些数据在区间 $0\leqslant x\leqslant 6$ 上的一个连续的图像.

x_i	0.0	1.0	2.0	3.0	4.0	5.0	6.0
y_i	2.000 0	2.159 2	3.169 7	5.433 2	9.141 1	14.406	21.303

4. 给定 $x_0<x_1$ 以及 x, 定义

$$\mu=\frac{x-x_0}{x_1-x_0}$$

设 $P_1(x)$ 是函数 $f(x)$ 在点 x_0 和点 x_1 处的插值. 证明

$$P_1(x) = f(x_0) + \mu\,[f(x_1) - f(x_0)]$$

这个公式曾是作数学函数表的线性插值最常用的办法.

5. 编写一个作线性插值并检验其精度的计算机程序. 输入 x_0 和 x_1, 然后利用 $y = e^x$(或者其他的常用函数) 产生数据值. 对于 $[x_0,\ x_1]$ 内外的各种 x 值, 计算 $P_1(x)$, e^x 和它们的差 $E(x) = e^x - P_1(x)$. 绘制 $E(x)$ 值的图像, 观察误差如何随着 x 的变化而变化.

6. 证明公式 (4.9) 可以简化为

$$P_2(x) = 4x^2 - 4x - 1$$

7. 给出插值数据 $\{(-2,-15),(-1,-8),(0,-3)\}$ 的二次多项式. 求出 $P_2(x)$ 位于 $x = 0$ 右端的第一个零点. 试问 $P_2(x)$ 有最大值吗?

8. 利用 (4.6), 求出插值下列数据的多项式 $P_2(x)$. 对于每一种情况都尽可能地简化结果.

(a) $\{(0,1),\ (1,2),\ (2,3)\}$;　　**(b)** $\{(0,1),\ (1,1),\ (2,1)\}$.

讨论你的计算结果.

9. 设结点$\{x_0,\ x_1,\ x_2\}$是等距的, 满足 $x_1 - x_0 = h = x_2 - x_1$. 回顾习题 4 中的记号, 定义 $\mu = (x - x_0)/h$. 设 $P_2(x)$ 是在结点 x_0, x_1, x_2 处插值 $f(x)$ 的次数 $\leqslant 2$ 的多项式. 证明

$$P_2(x) = f(x_0) + \mu[f(x_1) - f(x_0)] + \frac{\mu(\mu-1)}{2}\{[f(x_2) - f(x_1)] - [f(x_1) - f(x_0)]\}$$

这个公式曾是作数学函数表的二次插值最常用的方法. $f(x_{i+1}) - f(x_i)$ 称为 $f(x)$ 的向前差分, 习题 33 中将进一步讨论它们的用途.

10. 写出 $P_3(x)$ 完整的公式 (4.13), 其中包括全部四个多项式 $L_0(x)$,$L_1(x)$,$L_2(x)$, $L_3(x)$.

11. **(a)** 证明在次数 $\leqslant 3$ 的多项式中仅有一个多项式 $P_3(x)$ 满足下列插值条件

$$P_3(x_i) = y_i, \quad i = 0,1,2,3$$

其中 x_i 互不相同.

提示: 推广 (4.10) 及其随后给出的关于 $P_2(x)$ 唯一性的证明.

(b) 给出定理 4.1.5 中关于 $P_n(x)$ 唯一性的证明.

12. **(a)** 当 $n = 3$ 时, 解释为什么

$$L_0(x) + L_1(x) + L_2(x) + L_3(x) = 1$$

对所有的 x 都成立.

提示: 不必实际算出 (4.14) 中的函数 $L_i(x)$. 只要用适当选取的 $\{y_0, y_1, y_2, y_3\}$, 并利用 (4.13) 式即可.

(b) 把 (a) 中的结果推广到任意的次数 $n > 0$.

13. 对插值次数 $n = 3$ 继续做习题 12 中的工作, 解释为什么

$$x_0^j L_0(x) + x_1^j L_1(x) + x_2^j L_2(x) + x_3^j L_3(x) = x^j, \quad j \leqslant 3$$

$$x_0^j L_0(x) + x_1^j L_1(x) + x_2^j L_2(x) + x_3^j L_3(x) \neq x^j, \quad j > 3$$

14. **(a)** 记 $P_2^{(0,2)}(x)$ 为插值数据 $\{(x_0,y_0),(x_1,y_1),(x_2,y_2)\}$ 的二次多项式, 记 $P_2^{(1,3)}(x)$ 为插值数据 $\{(x_1,y_1),(x_2,y_2),(x_3,y_3)\}$ 的二次多项式. 最后记 $P_3(x)$ 为插值数据 $\{(x_0,y_0),(x_1,y_1),(x_2,y_2),(x_3,y_3)\}$ 的三次多项式. 证明

$$P_3(x)=\frac{(x_3-x)P_2^{(0,2)}(x)+(x-x_0)P_2^{(1,3)}(x)}{x_3-x_0}$$

(b) 如何把 (a) 推广到从 $n-1$ 次插值多项式构造出插值数据 $\{(x_0,y_0),\cdots,(x_n,y_n)\}$ 的多项式 $P_n(x)$?

15. 求次数 $\leqslant 3$ 的多项式 $P(x)$, 使得

$$P(0)=y_1,\ P(1)=y_2,\quad P'(0)=y_1',\ P'\ (1)=y_2'$$

其中 y_1, y_2, y_1', y_2' 是给定的常数. 所求出的多项式称为**三次埃尔米特插值多项式**(cubic Hermite interpolating polynomial)

提示: 把多项式写成

$$P(x)=y_1H_1(x)+y_1'H_2(x)+y_2H_3(x)+y_2'H_4(x)$$

类似于 (4.15), 三次多项式 $H_1(x)$, $H_2(x)$, $H_3(x)$, $H_4(x)$ 满足适当的性质. 例如, 选取三次多项式 $H_1(x)$ 满足

$$H_1(0)=1,H_1(1)=0,\quad H_1'(0)=0,H_1'(1)=0$$

16. 作为一个广义插值问题, 求二次多项式 $q(x)$, 使得

$$q(0)=-1,\quad q(1)=-1,\quad q'(1)=4$$

17. 给出下列插值问题的解: 求一个 $\deg(q)\leqslant 2$ 的多项式 $q(x)$, 使得

$$q(x_0)=y_0,\quad q(x_1)=y_1,\quad q'(x_1)=y_1'$$

其中 $x_0\neq x_1$.

提示: 把多项式写成 $q(x)=y_0M_0(x)+y_1M_1(x)+y_1'M_2(x)$, 其中 $\deg(M_i)\leqslant 2$, $i=0,1,2$, 并且每个 $M_i(x)$ 在点 x_0 和点 x_1 处满足适当的插值条件. 例如, $M_0(x)$ 应该满足

$$M_0(x_0)=1,\quad M_0(x_1)=M_0'(x_1)=0$$

18. 设 $x_0<x_1<x_2$ 且给定 $\{y_0,y_1',y_2\}$. 求 $\deg(q)\leqslant 2$ 的多项式 $q(x)$, 使得

$$q(x_0)=y_0,\quad q'(x_1)=y_1',\quad q(x_2)=y_2$$

为了使插值多项式 $q(x)$ 存在且唯一, 需要加上什么样的条件?

19. 设 $f(x)=\mathrm{e}^x$, 取 $x_0=0.85$, $x_1=0.87$, $x_2=0.89$. 利用 e^{x_0}, e^{x_1}, e^{x_2}, 计算 $f[x_0,x_1]$, $f[x_1,x_2]$, $f[x_0,x_1,x_2]$. 检查逼近 (4.19) 的精度.

20. 设 $z=(x_0+x_1)/2$, $h=(x_1-x_0)/2$. 为了分析逼近 (4.19), 考虑误差

$$E=f[x_0,x_1]-f'\left(\frac{x_0+x_1}{2}\right)=\frac{f(z+h)-f(z-h)}{2h}-f'(z)$$

使用 1.2 节中的泰勒定理关于 z 展开 $f(z+h)$ 和 $f(z-h)$, 展开式中包括变量 h 的次数 $\leqslant 3$ 的项. 利用这些展开式证明: 对小的 h 值, 有

$$E\approx\frac{h^2}{6}f'''(z)$$

因此, 当 x_1-x_0 相对较小时, (4.19) 是一个很好的逼近.

21. **(a)** 设 $f(x)$ 是一个 m 次多项式. 对于 $x\neq x_0$, 定义

$$g_1(x)=f[x_0,x]=\frac{f(x)-f(x_0)}{x-x_0}$$

证明 $g_1(x)$ 是一个 $m-1$ 次多项式. 这再次说明差商是导数的模拟.

提示: 分子 $f(x)-f(x_0)$ 是具有零点 $x=x_0$ 的 m 次多项式. 利用代数基本定理, 注意 x_0 是 $f(x)-f(x_0)$ 的根.

(b) 设 $f(x)$ 是 m 次多项式且 x_0,x_1,x_2 互不相同, 定义

$$g_2(x)=f[x_0,x_1,x]$$

证明 $g_2(x)$ 是一个 $m-2$ 次多项式.

22. 验证公式 (4.28), 从而证明 $f[x_0,x_1,x_2]$ 对节点的置换是对称的.

23. 利用 (4.29) 和得到 (4.30) 中所用的思想, 给出下列计算的公式.

(a) $f[x_0,x_0,x_1,x_1]$;　　**(b)** $f[x_0,x_0,x_0,x_1]$.

24. 对下列数据, 求 $f[x_0,x_1]$ 和 $f[x_0,x_1,x_2]$. 然后计算在 $x=0.15$ 处的线性插值和二次插值的值: $P_1(0.15)$ 和 $P_2(0.15)$.

n	x_n	$f(x_n)$
0	0.1	0.2
1	0.2	0.24
2	0.3	0.30

25. 设 $f(x)=1/(1+x)$, $x_0=0$, $x_1=1$, $x_2=2$. 计算差商 $f[x_0,x_1]$ 和 $f[x_0,x_1,x_2]$. 利用这些差商给出在给定的节点 $\{x_0,x_1,x_2\}$ 处插值 $f(x)$ 的二次多项式 $P_2(x)$. 画出区间 [0,2] 上误差 $f(x)-P_2(x)$ 的图像.

26. **(a)** 利用函数程序`divdif`, 对于 $f(x)=\mathrm{e}^x$, 计算差商 $D_0=f(x_0)$, $D_1=f[x_0,x_1]$, $\cdots$, $D_5=f[x_0,x_1,x_2,x_3,x_4,x_5]$, 其中 $x_0=0$, $x_1=0.2$, $x_2=0.4$, $\cdots$, $x_5=1.0$.

(b) 利用 (a) 的结果, 对于 $x=0.1,0.3,0.5$ 和 $j=1,\cdots,5$, 计算 $P_j(x)$. 把这些结果与 e^x 的真值进行比较.

27. 对 $f(x)=\tan^{-1}(x)$ 重复习题 26 中的工作.

28. 下列数据取自一个次数 $\leqslant 5$ 的多项式 $p(x)$. 试问这是什么样的多项式? 它的次数是多少?

x	-2	-1	0	1	2	3
$p(x)$	-5	1	1	1	7	25

29. 编写一个程序用牛顿公式 (4.35) 来检验高阶插值的精度. 对某个函数 $f(x)$ 和若干节点 $x_0, x_1, \cdots, x_6$, 用`divdif`给出差商

$$D_i = f[x_0, \cdots, x_i], \quad i = 0, 1, \cdots, 6$$

然后对于各个 x 值, 计算 $P_6(x)$, 并把它们与 $f(x)$ 的真值进行比较. 对于 $f(x) = \cos(x)$, 通过重新给出表 4-1 和表 4-2 中的一些结果来检验你的程序. 然后, 请你自行选择一个非多项式函数 $f(x)$ 重复上述过程.

30. (a) 在线性插值公式 (4.33) 中, 设 $x_1 - x_0 = h$ 且 $\mu = (x - x_0)/(x_1 - x_0) = (x - x_0)/h$. 证明 (4.33) 可化简为前面习题 4 中的公式.

(b) 在二次插值 (4.34) 中, 设 $x_1 - x_0 = x_2 - x_1 = h$ 且 $\mu = (x - x_0)/(x_1 - x_0)$. 证明 (4.34) 可化简为前面习题 9 中的公式.

31. 对某个整数 $n \geqslant 0$, 令 $f(x) = x^n$. 设 $x_0, x_1, \cdots, x_m$ 是 $m + 1$ 个相互不同的数. 当 $m = n$ 时 $f[x_0, x_1, \cdots, x_m]$ 是什么? 当 $m > n$ 时, $f[x_0, x_1, \cdots, x_m]$ 是什么?

32. 设 $f(x) = \mathrm{e}^x$. 对于任意的非负整数 m 以及任意一组互不相同的节点 $\{x_0, x_1, \cdots, x_m\}$, 证明 $f[x_0, x_1, \cdots, x_m] > 0$.

33. 当 $\{x_j\}$ 是等距节点时, 差商公式和插值多项式 $P_n(x)$ 都具有简化的形式. 对于 $j = 0, \pm 1, \pm 2, \cdots$, 设 $x_j = x_0 + jh$ 并设 $f_j \equiv f(x_j)$. 定义 $f(x)$ 的一阶**向前差分**(forward difference) 为

$$\Delta f(x_j) \equiv \Delta f_j = f_{j+1} - f_j$$

定义二阶向前差分为

$$\begin{aligned} \Delta^2 f(x_j) &= \Delta f_{j+1} - \Delta f_j = [f_{j+2} - f_{j+1}] - [f_{j+1} - f_j] \\ &= f_{j+2} - 2f_{j+1} + f_j \end{aligned}$$

通过下列公式递推定义高阶向前差分:

$$\Delta^k f_j = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j, \quad k \geqslant 2$$

证明

(a) $f[x_j, x_{j+1}] = \frac{1}{h} \Delta f_j$

(b) $f[x_j, x_{j+1}, x_{j+2}] = \frac{1}{2! h^2} \Delta^2 f_j$

(c) $f[x_j, x_{j+1}, \cdots, x_{j+k}] = \frac{1}{k! h^k} \Delta^k f_j$, $k \geqslant 1$

34. (a) 利用 (4.33) 和习题 33(a), 推导习题 4 的插值公式.

(b) 利用 (4.34) 和习题 33(b), 推导习题 9 的插值公式.

(c) 把习题 33(c) 用于 (4.35), 推广 (a) 和 (b). 利用**二项式系数**(binomial coeficient)

$$\binom{\mu}{k} = \frac{\mu(\mu - 1) \cdots (\mu - k + 1)}{k!}, \quad \binom{\mu}{0} = 1$$

证明

$$P_n(x)=\sum_{k=0}^{n}\begin{pmatrix}\mu\\k\end{pmatrix}\Delta^k f_0,\quad \mu=\frac{x-x_0}{h}$$

上式称为**插值多项式的牛顿向前差分形式**(Newton's forward difference form of the interpolation polynomial).

4.2 多项式插值的误差

对给定区间 $[a,b]$ 上定义的函数 $f(x)$, 设 $P_n(x)$ 表示在 $[a,b]$ 内的 $n+1$ 个节点 $x_0,x_1,\cdots,x_n$ 处插值 $f(x)$ 的次数 $\leqslant n$ 的多项式

$$P_n(x)=\sum_{j=0}^{n}f(x_j)L_j(x) \tag{4.40}$$

本节中, 我们将仔细考虑多项式插值的误差, 并给出当 x 和 n 变化时误差性态的更精确的信息. 我们首先给出一个误差公式.

定理 4.2.1 设 $n\geqslant 0$ 并设 $f(x)$ 在 $[a,b]$ 上有 $n+1$ 阶连续导数, 再设 $x_0,x_1,\cdots,x_n$ 是 $[a,b]$ 内互不相同的节点. 对 $a\leqslant x\leqslant b$, 有

$$f(x)-P_n(x)=\frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!}f^{(n+1)}(c_x) \tag{4.41}$$

其中 c_x 是介于 x 和 $x_0,x_1,\cdots,x_n$ 的最小值和最大值之间的某个数.

因为证明过程不含任何插值的额外有用的信息, 所以我们略去定理的证明. 习题 18 讨论 $n=1$ 时线性插值的误差公式的证明.

例 4.2.2 在区间 $[0,1]$ 上取 $f(x)=\mathrm{e}^x$. 考虑 $f(x)$ 利用节点 x_0 和 x_1 的线性插值的误差, 其中 $0\leqslant x_0\leqslant x_1\leqslant 1$. 由 (4.41) 知

$$\mathrm{e}^x-P_1(x)=\frac{(x-x_0)(x-x_1)}{2}\mathrm{e}^{c_x} \tag{4.42}$$

其中 c_x 是介于 x_0,x_1,x 的最小值和最大值之间的某个数. 在例题中, 假设 $x_0<x<x_1$. 因而可以看出线性插值的误差是负值, 把它改写为

$$\mathrm{e}^x-P_1(x)=-\frac{(x_1-x)(x-x_0)}{2}\mathrm{e}^{c_x}$$

倘若对于满足 $x_0<x<x_1$ 的 x, 把 e^{c_x} 近似地作为一个常数 (当区间 $[x_0,x_1]$ 很小时这个结论几乎正确), 则上式表明误差近似地为一个以 x_0 和 x_1 为根的二次多项式. 因为 $x_0\leqslant c_x\leqslant x_1$, 所以我们得到误差的上界和下界:

$$\frac{(x_1-x)(x-x_0)}{2}\mathrm{e}^{x_0}\leqslant|\mathrm{e}^x-P_1(x)|\leqslant\frac{(x_1-x)(x-x_0)}{2}\mathrm{e}^{x_1}$$

为了得到与 x 无关的误差界, 利用

$$\max_{x_0 \leqslant x \leqslant x_1} \frac{(x_1 - x)(x - x_0)}{2} = \frac{h^2}{8}, \quad h = x_1 - x_0 \tag{4.43}$$

上式易证, 只需注意到 $(x_1 - x)(x - x_0)$ 是以 x_0 和 x_1 为根的二次多项式, 因此它的最大值在两根之间. 把 $x = (x_0 + x_1)/2$ 代入上式得到 $h^2/8$.

因为在区间 $[0,1]$ 上 $\mathrm{e}^{x_1} \leqslant \mathrm{e}$, 所以我们得到与 x,x_0,x_1 无关的上界

$$|\mathrm{e}^x - P_1(x)| \leqslant \frac{h^2\mathrm{e}}{8}, \quad 0 \leqslant x_0 \leqslant x \leqslant x_1 \leqslant 1 \tag{4.44}$$

回顾 4.1 节中例 4.1.2, 取 $x = 0.826$ 以及 $h = 0.01$, 有

$$|\mathrm{e}^x - P_1(x)| \leqslant \frac{(0.01)^2(2.72)}{8} = 0.000\,034\,0 \tag{4.45}$$

实际误差是 $-0.000\,027\,6$, 它满足上述误差界. ■

例 4.2.3 再在区间 $[0,1]$ 上取 $f(x) = \mathrm{e}^x$. 考虑二次插值的误差, 则有

$$\mathrm{e}^x - P_2(x) = \frac{(x - x_0)(x - x_1)(x - x_2)}{6}\mathrm{e}^{c_x} \tag{4.46}$$

其中 c_x 是介于 x_0, x_1, x_2, x 的最小值和最大值之间的某个数. 假设 x_0,x_1,x_2 是等距节点, 并设 $h = x_1 - x_0 = x_2 - x_1$, 假设 $0 \leqslant x_0 < x < x_2 \leqslant 1$.
如前所述, 有

$$|e^x - P_2(x)| \leqslant \left|\frac{(x - x_0)(x - x_1)(x - x_2)}{6}\right| e^1 \tag{4.47}$$

为了进一步讨论, 利用

$$\max_{x_0 \leqslant x \leqslant x_2} \left|\frac{(x - x_0)(x - x_1)(x - x_2)}{6}\right| = \frac{h^3}{9\sqrt{3}} \tag{4.48}$$

按下面的方法, 用初等微积分可得到上述公式. 为了简化计算, 考虑特殊情况

$$w_2(x) = \frac{(x + h)x(x - h)}{6} = \frac{x^3 - xh^2}{6} \tag{4.49}$$

它的图像如图 4-4 所示, 3 次多项式 $w_2(x)$ 是 (4.48) 中多项式沿 x 轴的平移, 它们具有相同的形状和大小. 根据 (4.49), 要找点 x 满足

$$0 = w_2'(x) = \frac{3x^2 - h^2}{6}$$

于是 $x = \pm h/\sqrt{3}$, 并且

$$\left|w_2\left(\pm\frac{h}{\sqrt{3}}\right)\right| = \frac{h^3}{9\sqrt{3}}$$

(4.48) 得证. 把它用到 (4.47) 中, 得到

$$|e^x - P_2(x)| \leqslant \frac{h^3 e}{9\sqrt{3}} \approx 0.174h^3 \tag{4.50}$$

对 $h = 0.01$ 和 $0 \leqslant x \leqslant 1$, 有

$$|e^x - P_2(x)| \leqslant 1.74 \times 10^{-7} \tag{4.51}$$

请把上式与线性插值的结果 (4.45) 进行比较. ■

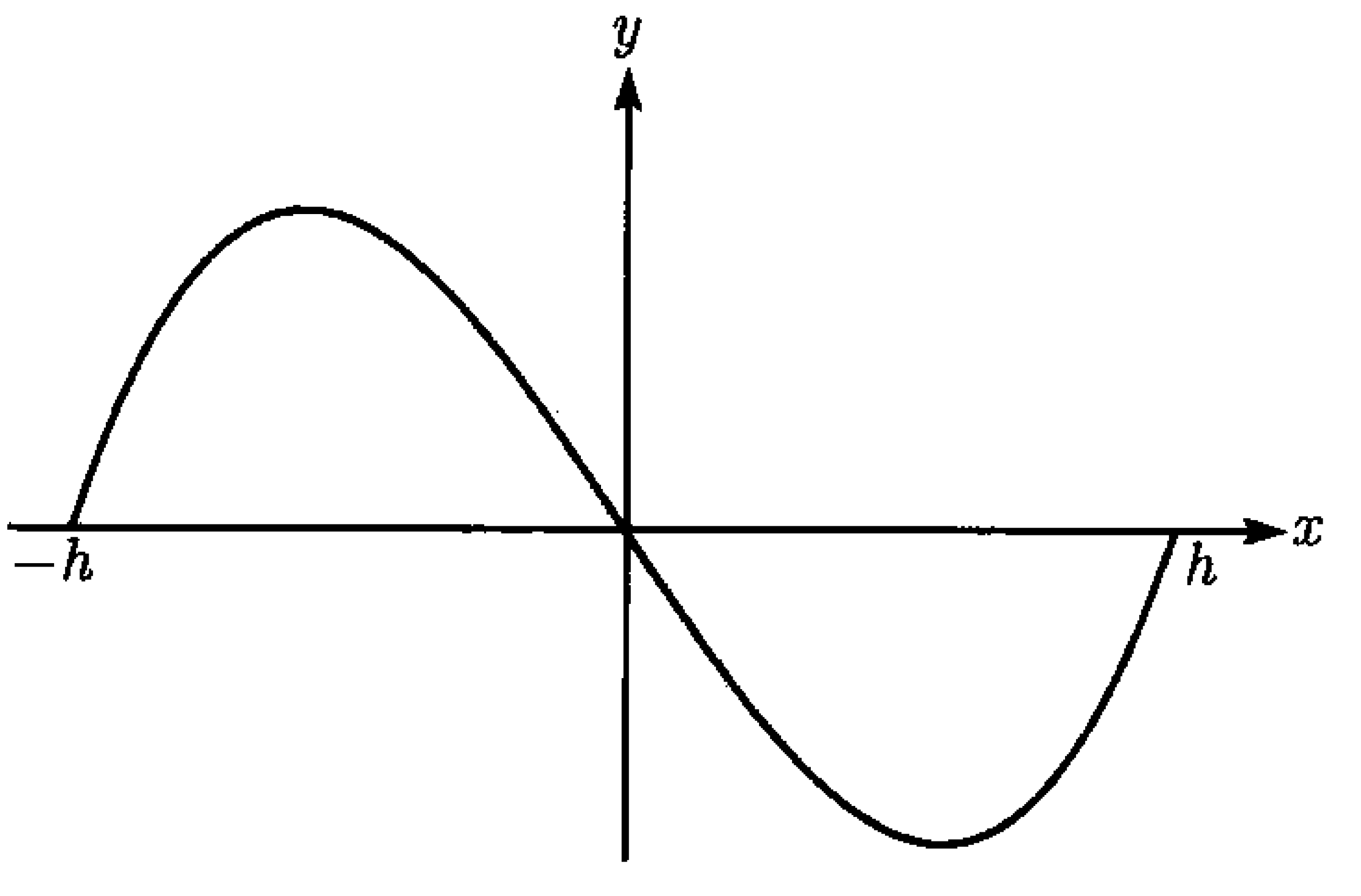


图 4-4 $y = w_2(x)$

4.2.1 另一个误差公式

构造在 $n+2$ 个节点 $x_0, x_1, \cdots, x_n, x_{n+1}$ 处 $f(x)$ 的插值公式 $P_{n+1}(x)$. 由牛顿差商形式知

$$P_{n+1}(x) = P_n(x) + (x - x_0)(x - x_1)\cdots(x - x_n)f[x_0, x_1, \cdots, x_n, x_{n+1}] \tag{4.52}$$

其中 $P_n(x)$ 在点 $x_0, x_1, \cdots, x_n$ 处插值 $f(x)$. 利用插值性质并且令 $x = x_{n+1}$, 有

$$f(x_{n+1}) = P_n(x_{n+1}) + (x_{n+1} - x_0)\cdots(x_{n+1} - x_n)f[x_0, x_1, \cdots, x_n, x_{n+1}]$$

把 x_{n+1} 看作不同于 $x_0, \cdots, x_n$ 的可变化的点, 并改记为 t. 把 P_n 项移到上述方程的左端, 得到

$$f(t) - P_n(t) = (t - x_0)(t - x_1)\cdots(t - x_n)f[x_0, x_1, \cdots, x_n, t] \tag{4.53}$$

这是 $P_n(t)$ 的插值误差. 如前面所述的那样, 如果 $f(x)$ 是一个充分可微函数, 则差商 $f[x_0, x_1, \cdots, x_n, t]$ 可扩展到 t 是一个节点的情形. 显然, 当 $t = x_0, x_1, \cdots, x_n$ 时, (4.53) 的右端是零.

对 $x = t$, 比较 (4.53) 与 (4.41), 可得插值误差的两个公式:

$$\frac{\Psi_n(t)}{(n+1)!}f^{(n+1)}(c_t) = \Psi_n(t)f[x_0, x_1, \cdots, x_n, t]$$

其中

$$\psi_n(t) = (t - x_0)(t - x_1)\cdots(t - x_n)$$

两端消去 $\psi_n(t)$, 得到

$$\frac{1}{(n+1)!}f^{(n+1)}(c_t) = f[x_0, x_1, \cdots, x_n, t] \tag{4.54}$$

其中 c_t 介于 $x_0, x_1, \cdots, x_n$, t 的最小值和最大值之间. 令 $m = n + 1$ 及 $t = x_m$, 则 (4.54) 变为

$$\frac{f^{(m)}(c)}{m!} = f\left[x_0, x_1, \cdots, x_{m-1}, x_m\right]$$

其中

$$\min\{x_0, \cdots, x_m\} \leqslant c \leqslant \max\{x_0, \cdots, x_m\}$$

这就证明了 4.1 节的定理 4.1.7.

我们通常使用公式 (4.41) 界定插值误差, 但有时候公式 (4.53) 更有用. 在数值积分方法的误差公式的推导过程中, 我们常使用公式 (4.53).

4.2.2 误差的性态

对于误差公式 (4.41) 和 (4.53) 来说, 多项式

$$\psi_n(x) = (x - x_0)\cdots(x - x_n) \tag{4.55}$$

是确定误差性态的最重要的量. 当节点 $x_0, x_1, \cdots, x_n$ 等距时, 我们将对 $x_0 \leqslant x \leqslant x_n$ 讨论误差的性态.

对于较大的 n, 比如 $n \geqslant 5$, 函数 $\psi_n(x)$ 的值穿过区间 $x_0 \leqslant x \leqslant x_n$ 时剧烈地变化. 区间 $[x_0, x_1]$ 和 $[x_{n-1}, x_n]$ 中的函数值变得远大于区间 $[x_0, x_n]$ 中部的函数值. 可以从理论上证明这一点, 但我们建议仅通过观察 $n = 6$ 时函数 $\psi_n(x)$ 的图像来说明这个结论. 如图 4-5 所示, 与区间 $[x_2, x_4]$ 上的函数值相比, 区间 $[x_0, x_1]$ 和 $[x_5, x_6]$ 上具有相对较大的函数值. 这种函数值之间的差距也会随着 n 的增大而增加.

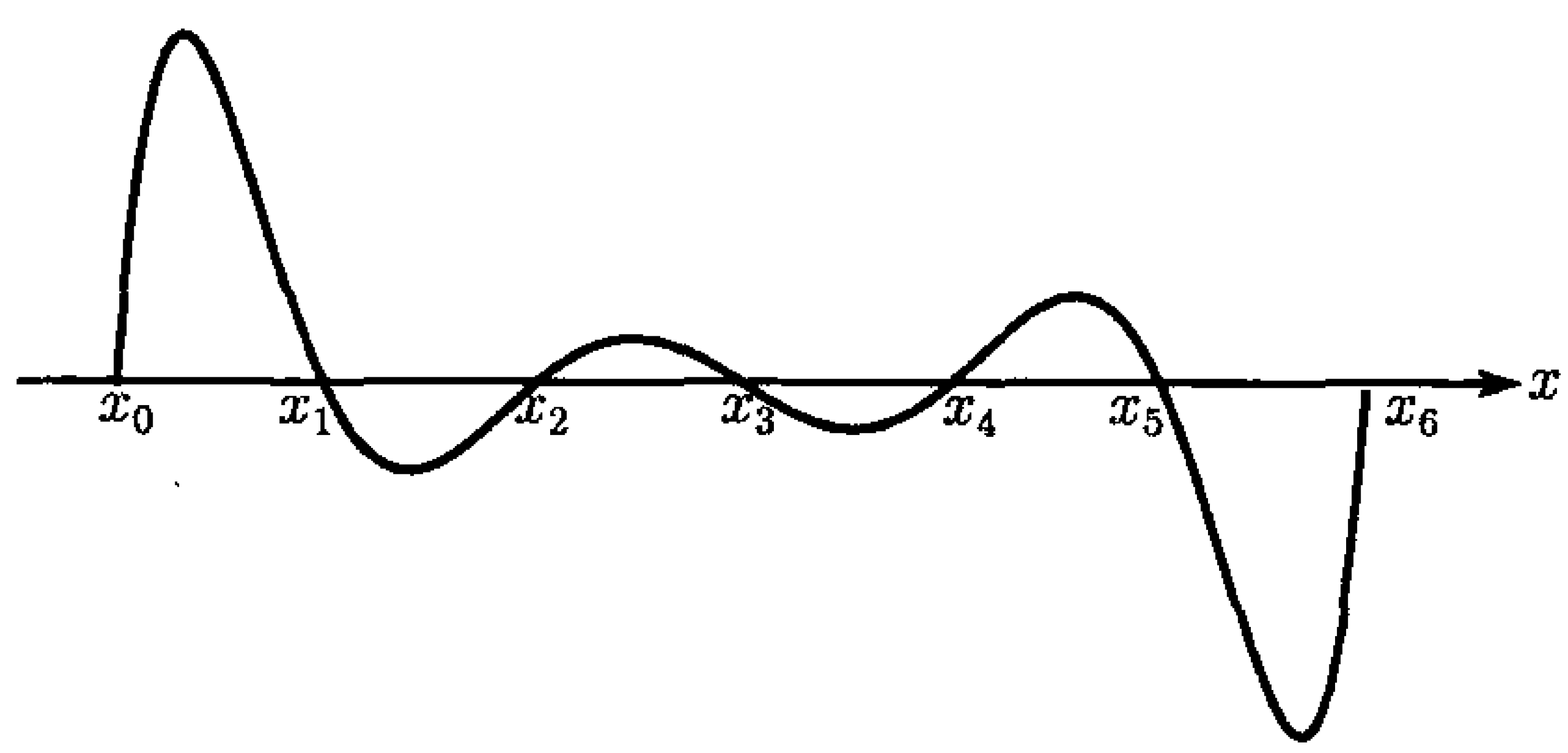


图 4-5 $y = \psi_6(x)$

当我们把 $\psi_n(x)$ 看作 $f(x)-P_n(x)$ 的误差公式 (4.41) 或 (4.53) 的组成部分时, 上面的讨论表明, 当 x 靠近节点集的两端时, 插值误差可能很大. 正是由于这些困难, 在实际插值问题中很少使用等距节点的高次多项式插值. 在 4.6 节中我们会看到, 通过选取适当的节点集可使函数的高次多项式插值变得非常有用.

例 4.2.4 设 $f(x)=\cos(x)$, $h=0.2$, $n=8$, 在点 $x=0.9$ 处插值.

情形 (i) $x_0=0.8$ 并且 $x_8=2.4$. 因此, $x=0.9$ 在第一个子区间 $[x_0,x_1]$ 中. 直接计算 $P_8(0.9)$ 得到

$$\cos(0.9)-P_8(0.9)\doteq -5.51\times 10^{-9}$$

情形 (ii) $x_0=0.2$ 并且 $x_8=1.8$. 因此, $x=0.9$ 在子区间 $[x_3,x_4]$ 中, 其中 x_4 是插值区间 $[x_0,x_8]$ 的中点. 直接计算得到

$$\cos(0.9)-P_8(0.9)\doteq 2.26\times 10^{-10}$$

该误差值是情形 (i) 中误差值的 1/24. ■

例 4.2.5 设 $f(x)=1/(1+x^2)$ 并且 $-5\leqslant x\leqslant 5$. 设 $n>0$ 是整数, 定义 $h=10/n$,

$$x_j=-5+jh,\quad j=0,1,2,\cdots,n$$

则对于区间 $[-5,5]$ 内的许多点 x, 可以证明当 $n\to\infty$ 时插值多项式序列 $\{P_n(x)\}$ 不收敛于 $f(x)$. 为了说明在这种情况下, $P_n(x)$ 对 $f(x)$ 的逼近很差, 我们在图 4-6 中给出了 $f(x)$ 和 $P_{10}(x)$ 的图像. 对于 $|x|\geqslant 4$, $P_n(x)$ 对 $f(x)$ 的背离程度随着 n 的增大而增加. ■

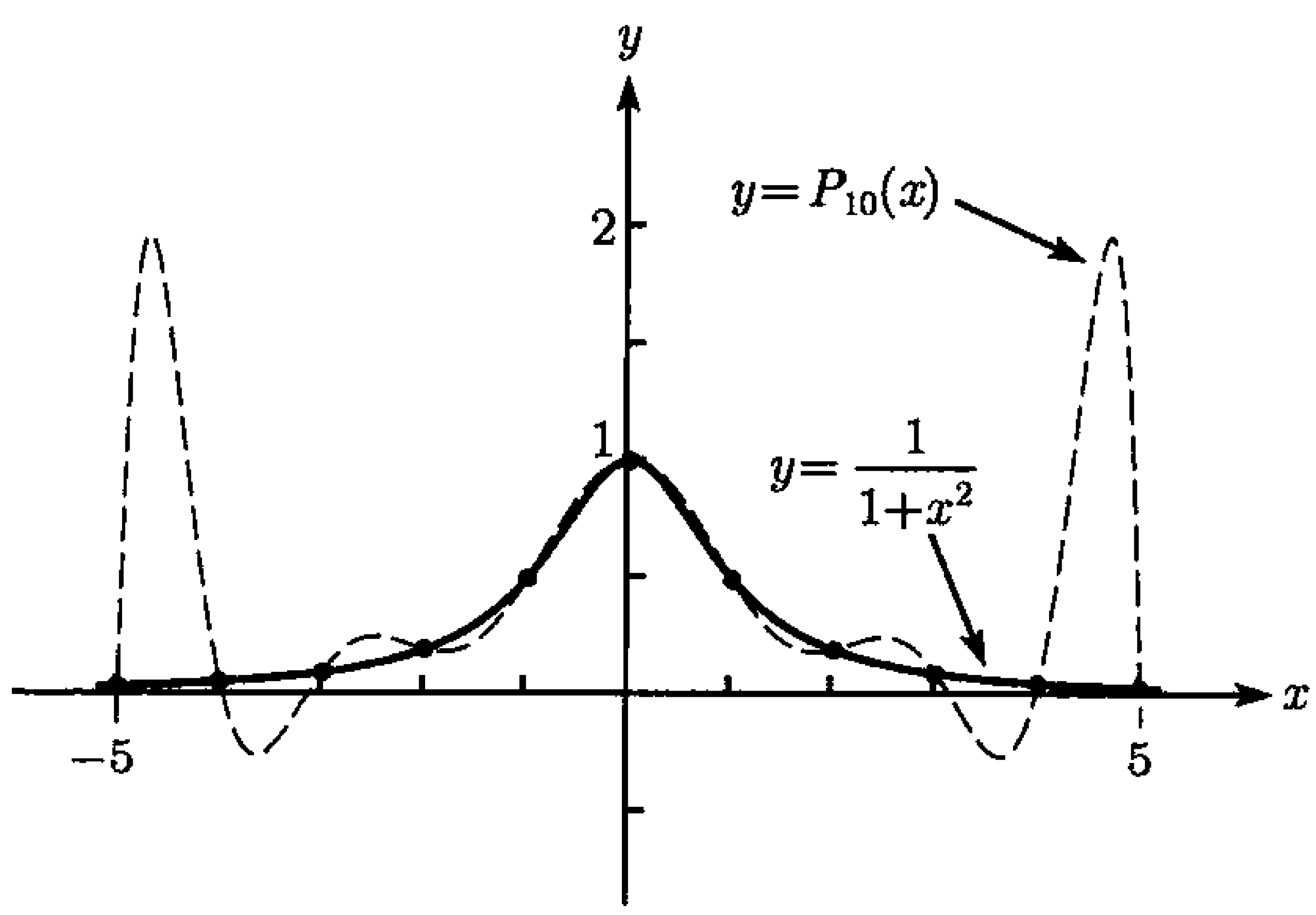


图 4-6 对 $1/(1+x^2)$ 的插值

习题

1. 根据函数值表作插值 $f(x)=\sin(x)$. 表中的函数值 f 在等距的节点 x 上给出, 其中 $0\leqslant x\leqslant 1.58$; 表中 x 的值按步长 $h=0.01$ 给出.

(a) 界定表中线性插值的误差 $f(x)-P_1(x)$. x 的值满足 $x_0<x<x_1$, 其中 x_0 和 x_1 是表中与 x 毗连的节点.

(b) 界定二次插值的误差 $f(x)-P_2(x)$ 的界. x 满足 $x_0<x<x_2$, 其中 x_0,x_1,x_2 是表中与 x 毗连的节点.

2. 在区间 $[0,0.8]$ 上对 $f(x)=\tan^{-1}(x)$ 重复习题 1 中的工作.

3. 对 $0\leqslant x\leqslant 1$ 和

$$f(x)=\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$$

重复习题 1 中的工作.

4. 假设对于 $0\leqslant x\leqslant 1.58$, $f(x)=\sin(x)$ 的函数值表是按步长 h 确定的 $\sin(x)$ 值所构成的.

(a) 如果在表中使用线性插值, 试问应该如何选取 h 值才能使插值误差小于 10^{-6}? 有关记号方面的假设参见习题 1(a).

(b) 如果在表中使用二次插值, 试问应该如何选取 h 值才能使插值误差小于 10^{-6}? 有关记号方面的假设参见习题 1(b).

5. 对于 $f(x)=\ln(x)$ 及 $1\leqslant x\leqslant 7$, 要求插值的误差容限是 5×10^{-6}. 重复习题 4 中的工作.

6. 对于 $f(x)=\tan^{-1}(x)$ 及 $0\leqslant x\leqslant 1$, 要求插值的误差容限是 5×10^{-6}. 重复习题 4 中的工作. 因为

$$\tan^{-1}(-x)=-\tan^{-1}(x)$$
$$\tan^{-1}\left(\frac{1}{x}\right)=\frac{\pi}{2}-\tan^{-1}(x),\quad x>0$$

从而对任意的 x, 可以直接计算 $\tan^{-1}(x)$.

7. 设 $1\leqslant x\leqslant 100$, 对 $x=0,h,2h,\cdots$, 构造 $f(x)=\sqrt{x}$ 的一个函数值表, 在表中使用线性插值时选择 h 使得误差的上界是 5×10^{-6}. 讨论并比较 $x=1$ 和 $x=100$ 附近的线性插值误差.

8. 设 $f(x)=x^4+\sqrt{2}x^3+\pi x$. 证实 $f[0,1,2,3,4]=f[0,1,\pi,\mathrm{e},-1]$ 是否成立.

9. 设多项式 $f(x)=a_0+a_1x+\cdots+a_nx^n$ 的次数不超过 n 并且 $\{x_0,x_1,\cdots,x_n\}$ 是互不相同的点. 试问 $f[x_0,x_1,\ldots,x_n]$ 的值是多少?

10. 界定 $f(x)=\mathrm{e}^x$ 对区间 $[0,1]$ 上等距节点的三次插值的误差界.

提示: 在区间 $-\frac{3}{2}h\leqslant x\leqslant\frac{3}{2}h$ 上, 用函数

$$\psi_3(x)=\left(x+\frac{3}{2}h\right)\left(x+\frac{1}{2}h\right)\left(x-\frac{1}{2}h\right)\left(x-\frac{3}{2}h\right)$$

的界替换函数

$$\psi_3(x)=(x-x_0)(x-x_1)(x-x_2)(x-x_3),\quad x_0\leqslant x\leqslant x_3$$

的界. 这里推广了在推导 (4.48) 的过程中使用的方法.

11. 使用节点

$$x_0=a-\frac{h}{\sqrt{3}},\quad x_1=a+\frac{h}{\sqrt{3}}$$

作函数 $f(x)$ 在区间 $[a-h, a+h]$ 上的线性插值, 其中 a 和 h 是实数且 $h>0$. 计算

$$\max_{a-h\leqslant x\leqslant a+h}|f(x)-P_1(x)|$$

的界. 以函数 $f(x)=\mathrm{e}^x$ 为例, 选择 a 和 h 的值并且画图解释你正在进行的计算. 这里讨论的线性插值可应用于求解某些微分方程和积分方程.

12. 当 n 增加并且 x 在区间 $[x_0, x_n]$ 上变化时, 为了直观地看出

$$\psi_n(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$$

的值的变化情况, 对 $0\leqslant x\leqslant n$, 画出特殊情况

$$\psi_n(x)=x(x-1)\cdots(x-n)$$

的图像. 对 $n=3,4,\cdots,8$ 的情形完成上面的工作.

13. 在区间 $[a,b]$ 上, 令 $h=(b-a)/n$, 其中整数 $n>0$. 定义等距节点为

$$x_j=a+jh,\quad j=0,1,\cdots,n$$

因此, $x_0=a, x_1=a+h,\cdots,x_n=a+nh=b$. 考虑多项式

$$\psi_n(x)=(x-x_0)(x-x_1)\cdots(x-x_n)$$

并且证明

$$|\psi_n(x)|\leqslant n!h^{n+1},\quad a\leqslant x\leqslant b \tag{*}$$

提示: 首先考虑 $n=2$ 或 $n=3$ 这样的低阶情况. 再分别考虑各种情况: $x_0<x<x_1$, $x_1<x<x_2,\cdots,x_{n-1}<x<x_n$. 对每种情况, 用 h 的倍数界定不同的因子 $x-x_j$.

14. 设 $P_n(x)$ 是区间 $[a,b]=[0,1]$ 上利用等距节点 (其定义见习题 13) 插值 $f(x)=\mathrm{e}^x$ 的 n 次多项式. 利用习题 13 中给出的主要结果 (*) 式, 证明

$$\text{当}n\to\infty\text{时},\quad \max_{0\leqslant x\leqslant 1}|\mathrm{e}^x-P_n(x)|\to 0$$

15. 在区间 $[0,\pi]$ 上对 $f(x)=\sin x$ 重复习题 14 中的工作.

16. 对给定区间 $[a,b]$ 的一个等距剖分, 计算函数

$$f(x)=\frac{1}{2+\cos x}$$

的插值多项式 $P_n(x)$. 对不同的 n, 研究区间 $[0,2\pi]$ 和 $[-\pi,\pi]$ 上的插值多项式及其误差. 对 $n=10$, 20, 30, 40 的情形完成上面的工作. 并观察其误差.

提示: 图像, 特别是插值误差的图像会有助于习题的解答. 当 $n>40$ 时, 即使双精度运算也已不够用.

17. 研究利用等距节点 (其定义见习题 13) 插值函数 $f(x)=\tan^{-1}x$ 的误差. 对于 $a=1,2,3$, 4 的情形, 分别给出 $[-a,a]$ 上相应的结论. 对于 $n=3,7,11,15,19$ 的情形构造 $P_n(x)$. 对 a 和 n 的每个值, 估计区间 $[-a,a]$ 上 $|f(x)-P_n(x)|$ 的极大值并列表给出结果. 并对尽可能多的情形绘出其误差的图像. 评注你所得到的结果.

18. 考虑线性插值的误差公式

$$f(x) - P_1(x) = \frac{(x-x_0)(x-x_1)}{2} f''(c)$$

的证明过程, 其中, $\min\{x_0, x_1, x\} \leqslant c \leqslant \max\{x_0, x_1, x\}$. 由 $P_1(x)$ 的结构知, 如果 $x = x_0$ 或者 $x = x_1$, 则误差公式显然成立. 因而, 我们只考虑 $x \neq x_0$, 且 $x \neq x_1$ 的情形. 引入辅助函数

$$E(t) = f(t) - P_1(t)$$

以及

$$G(t) = E(t) - \frac{(t-x_0)(t-x_1)}{(x-x_0)(x-x_1)} E(x)$$

其中 t 是变量, 而 x 是固定的.

(a) 证明 $G(x_0) = G(x_1) = G(x) = 0$.

(b) 利用罗尔定理 (Roll's theorem) 或者中值定理, 证明 $G'(t)$ 至少有两个零点; 然后证明 $G''(t)$ 至少有一个根, 把这个根记为 c.

(c) 计算 $E''(t)$ 和 $G''(t)$. 然后计算 $G''(c)$ 并且求解 $E(x)$, 以得到误差公式. 上述过程可推广成对任意的 n, 定理 4.2.1 的证明.

4.3 插值样条函数

为了介绍样条函数的定义和用途, 我们首先考虑表 4-3 中数据的插值问题. 最简单的插值方法就是用直线段把节点连接起来, 得到的图像如图 4-7 所示. 这称为**分段线性插值**(piecewise linear interpolation), 相应的插值函数记为 $l(x)$. 它与数据相符, 缺点是不具有光滑的图像. 大部分数据来自光滑曲线, 其图像没有 $y = l(x)$ 的那些隅角. 因此, 我们通常要构造插值给定数据点的一条光滑曲线, 同时它又顺着 $y = l(x)$ 的形状.

表 4-3 插值数据点

x	0	1	2	2.5	3	3.5	4
y	2.5	0.5	0.5	1.5	1.5	1.125	0

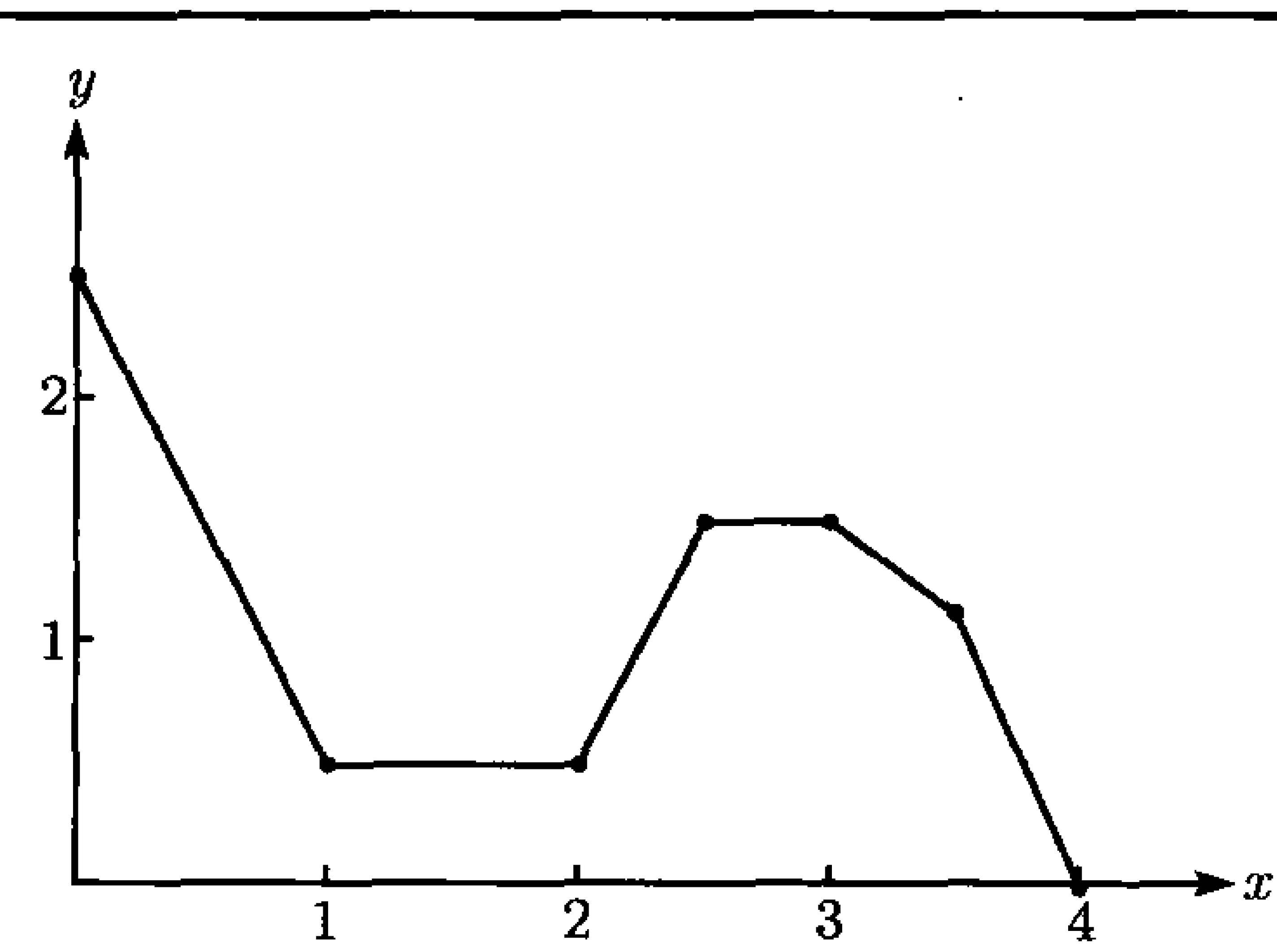


图 4-7 $y = l(x)$: 分段线性插值

第二种可选择的插值是用多项式插值. 表 4-3 中有 7 组数据点, 所以我们考虑 6 次插值多项式 $P_6(x)$, 它的图像如图 4-8 所示 (注意纵坐标尺度的变化), 明显地不同于 $y = l(x)$ 的图像. 尽管它是一条光滑曲线, 但在某些插值节点之间, 例如在 $0 \leqslant x \leqslant 1$ 上, 它与 $y = l(x)$ 的形状完全不同.

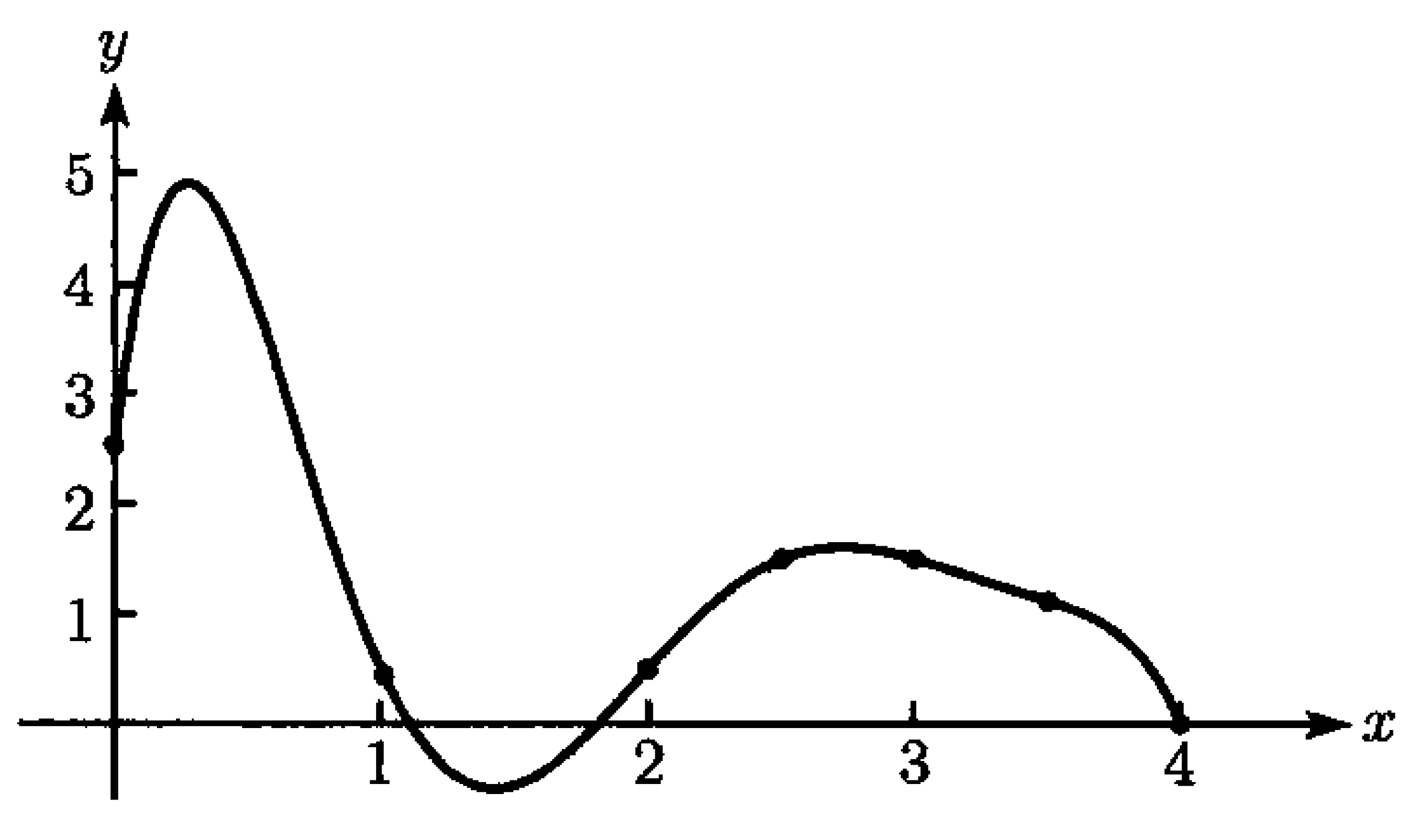


图 4-8　$y = P_6(x)$: 多项式插值

第三种可选择的插值是逐次用二次插值多项式连接表 4-3 中的数据点, 把区间 $0 \leqslant x \leqslant 4$ 上的这个函数记为 $q(x)$, 其图形如图 4-9 所示. 在每个子区间 $[0, 2]$, $[2, 3]$ 和 $[3, 4]$ 上, $q(x)$ 分别是插值这些子区间上数据点的二次多项式. 与 $l(x)$ 相比, $q(x)$ 的曲线光滑一些, 同时它的形状比 $y = P_6(x)$ 更靠近 $y = l(x)$ 一些. 不过, 函数的图形在点 $x = 2$ 和 $x = 3$ 处仍然具有 "隅角", 其导数 $q'(x)$ 在这些点处是不连续的. 我们希望寻找一个插值函数, 它是光滑的, 并且与 $y = l(x)$ 的图形相比, 在节点之间插值函数的变化不要太大. 然而, 就很多应用问题来说, 插值函数 $q(x)$ 可能是完全合适的, 例如, 利用给定数据点的数值积分就是这样的问题.

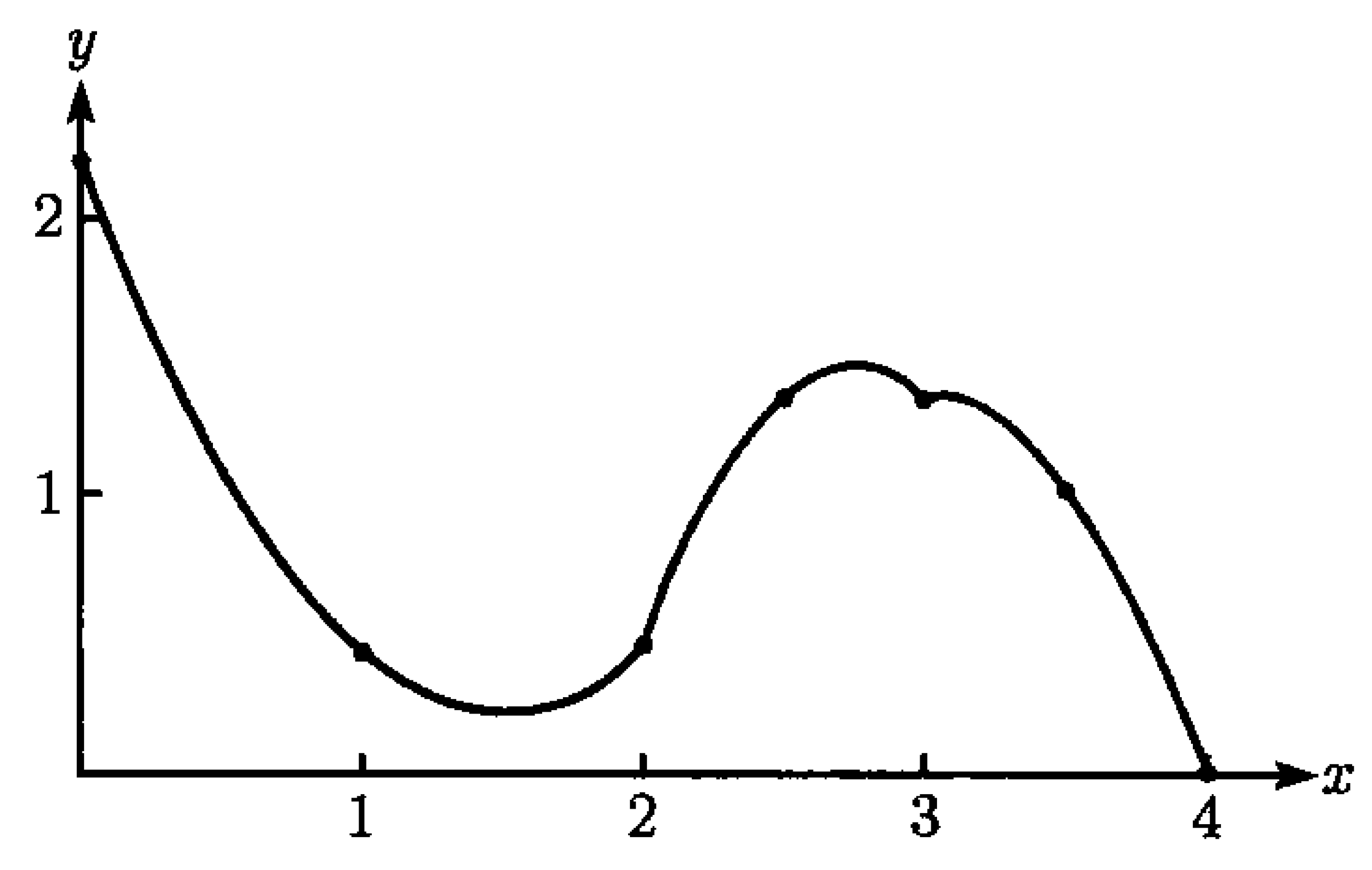


图 4-9　$y = q(x)$: 分段二次插值

4.3.1　样条插值

为了提出更一般的问题, 假设给定 n 个数据点 (x_i, y_i), $i = 1, \cdots, n$. 为了简单起见, 假设

$$x_1 < x_2 < \cdots < x_n \tag{4.56}$$

其中 $a = x_1$, $b = x_n$. 我们要寻找定义在 $[a, b]$ 上的一个函数 $s(x)$, 它插值数据

$$s(x_i) = y_i, \quad i = 1, \cdots, n \tag{4.57}$$

对于 $s(x)$ 的光滑性, 我们要求 $s'(x)$ 和 $s''(x)$ 都是连续的. 此外, 希望函数的曲线能顺着由连接数据点 (x_i, y_i) 得到的分段线性函数给出的基本形状, 如图 4-7 所示. 可以做到这一点的标准方法是要求导数 $s'(x)$ 在节点之间的变化不要太快. 这又要求二阶导数 $s''(x)$ 尽可能地小, 更具体地说, 要求

$$\int_a^b [s''(x)]^2 \mathrm{d}x \tag{4.58}$$

尽可能地小. 对于数据 $\{(x_i, y_i)\}_{i=1}^n$ 的一个光滑且保持形状的插值函数来说, 这也许不是一个完美的数学实现, 但从视觉上它通常给出一个非常好的插值函数.

这个问题有唯一解 $s(x)$, 并且它满足下列条件:

S1. 对于 $j = 2, 3, \cdots, n$, $s(x)$ 是每个子区间 $[x_{j-1}, x_j]$ 上次数 $\leqslant 3$ 的多项式;

S2. 对于 $a \leqslant x \leqslant b$, $s(x)$, $s'(x)$ 和 $s''(x)$ 都是连续的;

S3. $s''(x_1) = s''(x_n) = 0$.

函数 $s(x)$ 称为插值数据 $\{(x_i, y_i)\}$ 的**自然三次样条函数**(natural cube spline function). 我们将给出构造 $s(x)$ 的方法, 并把它应用于表 4-3 中的数据. 然后, 我们再对三次样条函数及其插值作一般性的讨论.

4.3.2 插值的自然三次样条的构造

引入下列参数:

$$M_i \equiv s''(x_i), \quad i = 1, 2, \cdots, n \tag{4.59}$$

我们将利用这些参数来给出 $s(x)$ 的表达式, 然后给出一个可以计算值 M_i 的线性方程组.

因为在每个区间 $[x_{j-1}, x_j]$ 上 $s(x)$ 是三次多项式, 所以在这些区间上 $s''(x)$ 是线性的. 线性函数由它在两个点处的函数值确定, 利用

$$s''(x_{j-1}) = M_{j-1}, \quad s''(x_j) = M_j \tag{4.60}$$

则

$$s''(x) = \frac{(x_j - x)M_{j-1} + (x - x_{j-1})M_j}{x_j - x_{j-1}}, \quad x_{j-1} \leqslant x \leqslant x_j \tag{4.61}$$

在区间 $[x_{j-1}, x_j]$ 上对 $s''(x)$ 积分两次并且应用插值条件

$$s(x_{j-1}) = y_{j-1}, \quad s(x_j) = y_j \tag{4.62}$$

经过一系列推导之后, 对于 $x_{j-1} \leqslant x \leqslant x_j$, 得到一个三次多项式

$$\begin{aligned} s(x) = &\frac{(x_j - x)^3 M_{j-1} + (x - x_{j-1})^3 M_j}{6(x_j - x_{j-1})} + \frac{(x_j - x)y_{j-1} + (x - x_{j-1})y_j}{x_j - x_{j-1}} \\ &- \frac{1}{6}(x_j - x_{j-1})[(x_j - x)M_{j-1} + (x - x_{j-1})M_j] \end{aligned} \tag{4.63}$$

可以直接验证这个公式的二阶导数产生 (4.61), 通过直接代入可知 (4.63) 满足插值条件 (4.62).

把公式 (4.63) 应用到区间 $[x_1, x_2], \cdots, [x_{n-1}, x_n]$ 上. 因为满足插值条件 $s(x_j) = y_j$, 所以公式在相邻区间 $[x_{j-1}, x_j]$ 和 $[x_j, x_{j+1}]$ 的公共点 $x = x_j$ 处是一致的. 这说明了 $s(x)$ 在整个区间 $[a, b]$ 上是连续的. 类似地, $s''(x)$ 的公式 (4.61) 也表明了它在 $[a, b]$ 上的连续性.

为了保证 $s'(x)$ 在 $[a, b]$ 上连续, 对于 $j = 2, 3, \cdots, n-1$, 要求 $s'(x)$ 在 $[x_{j-1}, x_j]$ 和 $[x_j, x_{j+1}]$ 上的公式在区间的公共点 $x = x_j$ 处有相同的值. 经过大量的化简工作之后, 可导出下列线性方程组

$$\begin{aligned} &\frac{x_j - x_{j-1}}{6} M_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} M_j + \frac{x_{j+1} - x_j}{6} M_{j+1} \\ = &\frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}, \quad j = 2, 3, \cdots, n-1 \end{aligned} \tag{4.64}$$

将这 $n-2$ 个方程结合前面的条件 (S3)

$$M_1 = M_n = 0 \tag{4.65}$$

可求出 $M_1, \cdots, M_n$ 的值, 从而给出插值函数 $s(x)$. 线性方程组 (4.64) 是一个**三对角方程组**(tridiagonal system), 有专门的方法求其解. 6.4 节给出了其中一种方法, 大部分计算机中心都有求解三对角方程组的专用程序.

例 4.3.1　我们来构造插值下列数据

$$\left\{(1,1), \left(2, \frac{1}{2}\right), \left(3, \frac{1}{3}\right), \left(4, \frac{1}{4}\right)\right\} \tag{4.66}$$

的自然三次样条. 点的个数 $n = 4$, 所有的 $x_j - x_{j-1} = 1$. 方程组 (4.64) 变为

$$\begin{aligned} \frac{1}{6}M_1 + \frac{2}{3}M_2 + \frac{1}{6}M_3 = \frac{1}{3} \\ \frac{1}{6}M_2 + \frac{2}{3}M_3 + \frac{1}{6}M_4 = \frac{1}{12} \end{aligned} \tag{4.67}$$

结合 (4.65), 得到

$$M_2 = \frac{1}{2}, \quad M_3 = 0$$

把它们代入 (4.63), 得到

$$s(x)=\begin{cases}\dfrac{1}{12}x^3-\dfrac{1}{4}x^2-\dfrac{1}{3}x+\dfrac{3}{2}, & 1\leqslant x\leqslant 2,\\ -\dfrac{1}{12}x^3+\dfrac{3}{4}x^2-\dfrac{7}{3}x+\dfrac{17}{6}, & 2\leqslant x\leqslant 3,\\ -\dfrac{1}{12}x+\dfrac{7}{12}, & 3\leqslant x\leqslant 4\end{cases} \tag{4.68}$$

我们把计算 $s'(x)$ 和 $s''(x)$ 并检验其连续性的工作留作习题. ■

例 4.3.2 考虑插值表 4-3 中数据的自然三次样条. 因为点的个数 $n=7$, 所以方程组 (4.64) 含 5 个方程. 所得函数 $s(x)$ 的图像如图 4-10 所示; 为了便于比较, 我们在图中也给出了图 4-7 的分段线性插值 $y=l(x)$ 的图像. $y=s(x)$ 的图像与图 4-9 中 $q(x)$ 的图像形状相似. 不同于 $y=q(x)$ 在点 $x=2$ 和 $x=3$ 处那样, $s(x)$ 的图形不再有隅角或者说它具有连续变化的斜率. 对于表 4-3 中的数据, $s(x)$ 的图像看上去似乎是比 $q(x)$ 更好的插值函数. ■

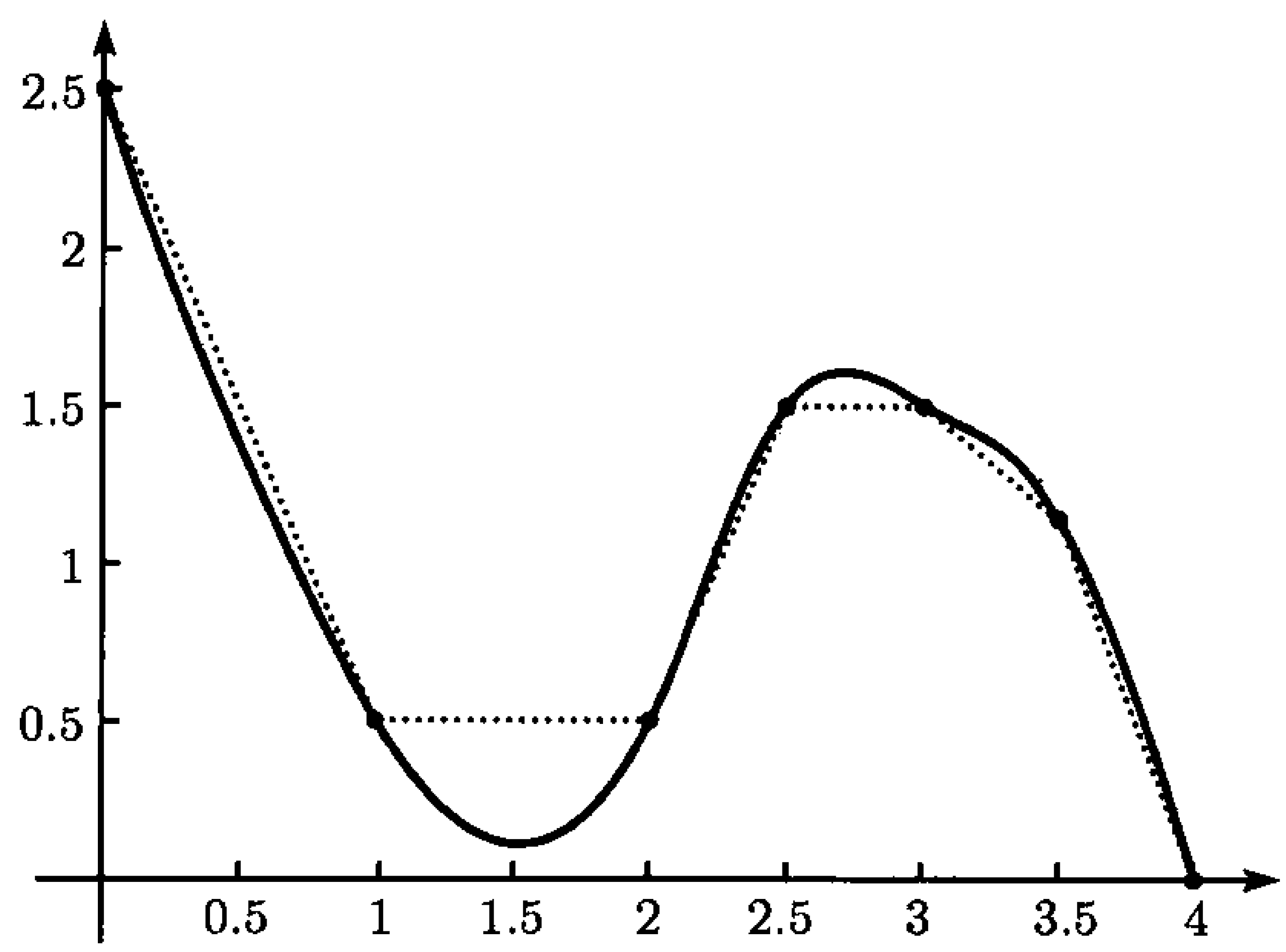


图 4-10 自然三次样条插值 (实线) 和分段线性插值 (虚线)

4.3.3 其他插值样条函数

到现在为止, 我们还没有考虑插值样条 $s(x)$ 的精度. 如果只知道一些数据点, 而我们只需要一条看上去适当的光滑曲线, 则用自然三次样条函数做插值是令人满意的. 但是, 我们通常要用样条去插值一个已知函数, 因此我们也对插值的精度感兴趣.

给定 $[a,b]$ 上的函数 $f(x)$. 我们考虑在 x 的等距值上执行 $f(x)$ 的插值情况. 设 $n>1$, 取

$$h=\frac{b-a}{n-1},\quad x_j=a+(j-1)h,\quad j=1.2,\cdots,n \tag{4.69}$$

设 $s_n(x)$ 是在 $x_1, \cdots, x_n$ 处插值 $f(x)$ 的自然三次样条. 则可以证明

$$\max_{a \leqslant x \leqslant b} |f(x) - s_n(x)| \leqslant ch^2 \tag{4.70}$$

其中 c 依赖于 $f''(a)$, $f''(b)$ 和 $\max_{a\leqslant x\leqslant b}\left|f^{(4)}(x)\right|$. 逼近函数 $s_n(x)$ 不能更快地收敛 (即, 有一个 h 的较高幂次的误差界) 的主要原因是在 $x = a$ 及 $x = b$ 处 $f''(x)$ 一般是非零的, 而根据定义知 $s_n''(a) = s_n''(b) = 0$. 对于满足条件 $f''(a) = f''(b) = 0$ 的函数 $f(x)$, (4.70) 式右端项可用 ch^4 代替.

为了改进 $s_n(x)$ 的逼近精度, 可以考虑插值 $f(x)$ 的其他三次样条函数 $s(x)$. 回到由 (4.56) 给出的一般节点定义, 如果

(1) 在每个子区间 $[x_{j-1}, x_j]$ 上 $s(x)$ 是三次多项式;

(2) 在 $[a, b]$ 上 $s(x)$, $s'(x)$ 和 $s''(x)$ 都是连续的.

那么我们称 $s(x)$ 是 $[a, b]$ 上的**三次样条**(cubic spline).

一般来说, 三次样条是便于使用的相当光滑的函数. 在过去的二十年间, 它被广泛地用于计算机制图以及应用数学和统计学的许多领域.

如果再要求 $s(x)$ 满足 (4.57) 的插值条件

$$s(x_i) = y_i, \quad i = 1, \cdots, n$$

则表示公式 (4.63) 和三对角方程组 (4.64) 仍然成立. 这个方程组含有 $n-2$ 个方程以及 n 个未知量 $M_1, \cdots, M_n$. 用其他条件代替端点条件 (4.65), 可以得到不同的三次样条插值.

如果数据来自 $f(x)$ 的函数值:

$$y_i = f(x_i), \quad i = 1, \cdots, n$$

则我们可以选择 $s(x)$ 的端点条件 (或者边界条件) 以便得到 $f(x)$ 的更好的逼近. 如果可能的话, 要求

$$s'(x_1) = f'(x_1), \quad s'(x_n) = f'(x_n) \tag{4.71}$$

或者

$$s''(x_1) = f''(x_1), \quad s''(x_n) = f''(x_n) \tag{4.72}$$

当结合 (4.63) 和 (4.64) 时, 上述两组条件中的任一组可引导出唯一的插值样条 $s(x)$, 其中 $s(x)$ 与所选的端点条件相关. 在这两种情况下, (4.70) 的右端项都可由 ch^4 代替, 其中 c 与 $\max_{a\leqslant x\leqslant b}\left|f^{(4)}(x)\right|$ 相关.

如果 $f(x)$ 的导数是未知的, 则也可用额外的插值条件来保证 (4.70) 的误差界与 h^4 成比例. 特别地, 假设

$$x_1 < z_1 < x_2, \quad x_{n-1} < z_2 < x_n$$

并且假设 $f(z_1)$ 和 $f(z_2)$ 都是已知的. 那么利用 (4.63) 中 $s(x)$ 的公式以及

$$s(z_1) = f(z_1), \quad s(z_2) = f(z_2) \tag{4.73}$$

这将在方程组 (4.64) 中添加两个新方程, 一个是关于 M_1 和 M_2 的方程, 另一个是关于 M_{n-1} 和 M_n 的方程. 这种形式的样条一般比插值的自然三次样条更可取, 而且构造它的过程几乎同样地容易. MATLAB 中实现的样条插值就是这种形式, 下面我们进一步讨论它. 用这种方法构成的样条形式称为满足**非节点插值边界条件**(not-a-knot interpolation boundary conditions). 图 4-11 给出 $l(x)$ 的图像, 以及表 4-3 中给出数据 "非节点" 插值的三次样条的图像 ($z_1 = 1$, $z_2 = 3.5$). 比较图 4-11 和图 4-10.

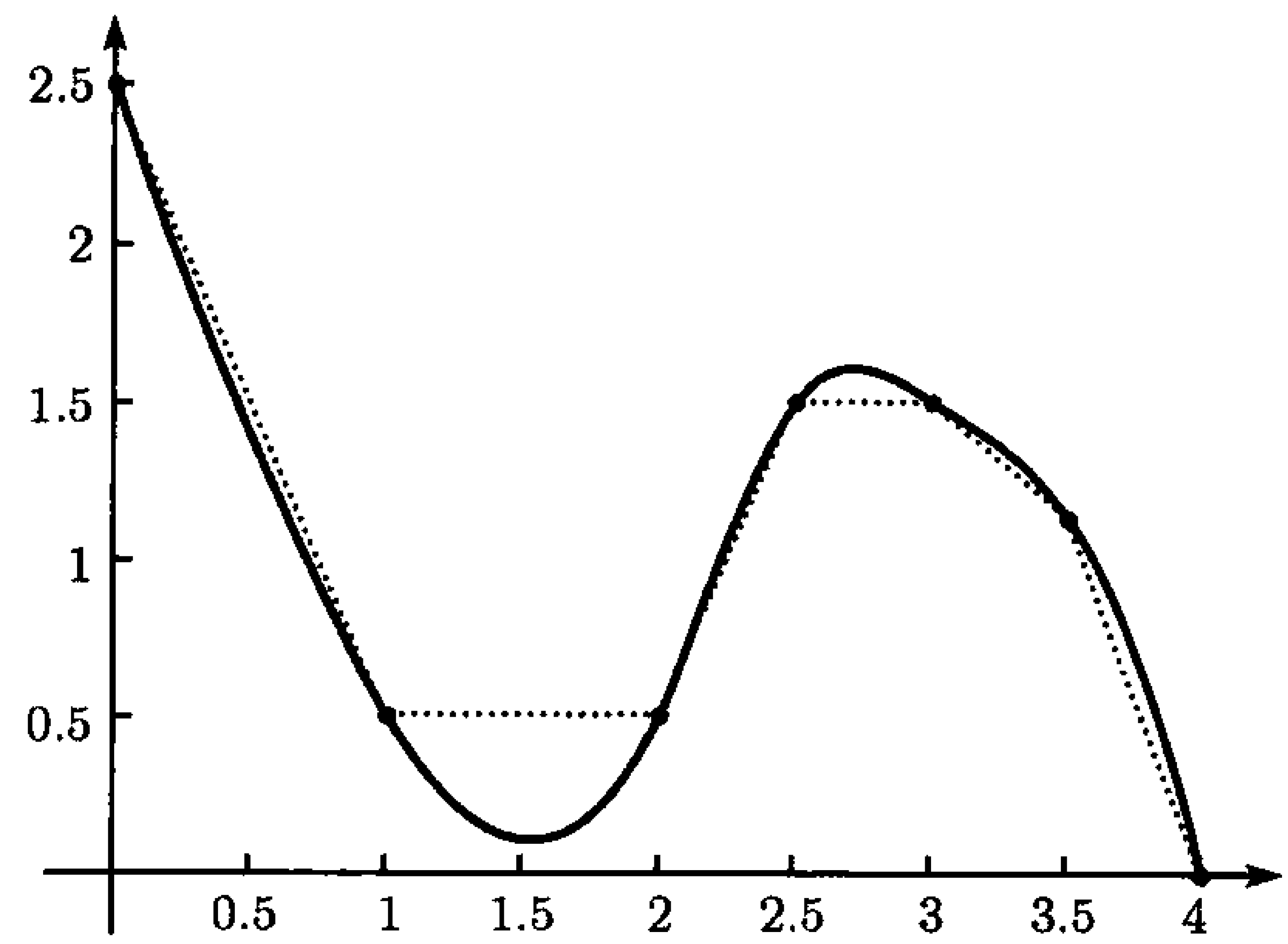


图 4-11　非节点插值的样条 (实线) 和分段线性插值 (虚线)

因为插值的三次样条函数比较光滑 (有两阶连续的导数), 它又不像高次多项式插值那样有时会出现剧烈振荡, 并且也比较容易利用计算机进行计算, 所以插值三次样条函数已成为解析表示数据的常用方法. 它们并不替代多项式, 而是多项式的一种非常有用的推广.

大部分高等学校的计算机中心都有一些关于三次样条的程序, 用于数值积分、数值微分、插值以及数据的曲线拟合等. 插值程序一般会考虑到种种端点条件, 如 (4.65) 以及 (4.71)~(4.73) 中给出的那样. 我们还是建议同学去使用这些软件包而不是自行编写程序.

4.3.4 MATLAB 程序spline

标准的 MATLAB 软件包含函数spline. 它有几个调用序列, 标准的调用序列是

$$\mathtt{y = spline(x_nodes, y_nodes, x)}$$

它产生三次样条函数 $s(x)$, 函数图像通过点 $\{(\xi_i, \eta_i) | i = 1, \cdots, n\}$, 其中

$$(\xi_i, \eta_i) = (\texttt{x_nodes(i)}, \texttt{y_nodes(i)})$$

其中 n 是x_nodes(或者y_nodes) 的长度. 使用的是 (4.73) 中的非节点插值条件. 点 (ξ_2, η_2) 是 (4.73) 中的点 $(z_1, f(z_1))$, 而点 (ξ_{n-1}, η_{n-1}) 是点 $(z_2, f(z_2))$. 得到三次样条 $s(x)$ 之后, 计算 $s(x)$ 在给定的横坐标 x 处的值, 从spline中输出这些值并把它们贮存在 y 中.

如果使用语句

$$\texttt{pp} = \texttt{spline(x_nodes, y_nodes)}$$

$$[\texttt{breaks}, \texttt{coefs}, \texttt{l}, \texttt{k}, \texttt{d}] = \texttt{unmkpp(pp)}$$

则可以得到x_nodes中毗连的横坐标所确定的每个子区间上 $s(x)$ 的三次多项式表达式的系数. 可以从 MATLAB 的帮助(help) 指令中得到关于这些指令的更详细的信息.

例 4.3.3 在区间 $[a, b] = [0, 1]$ 上逼近函数 $f(x) = e^x$. 对于 $n > 0$, 定义 $h = 1/n$. 我们选择两种不同的插值节点. 首先, 我们选取节点

$$x_1 = 0,\ x_2 = h,\ x_3 = 2h, \cdots, x_{n+1} = nh = 1$$

把这个选择称为 "Choice#1". 利用spline, 产生 $f(x)$ 的三次样条插值 $s_{n,1}(x)$. 对相应的非节点插值条件, 节点 x_2 和节点 x_n 分别是 (4.73) 中的点 z_1 和点 z_2. 对一般的光滑函数 $f(x)$, 通常是在逼近函数的区间端点附近的误差值 $f(x) - s_{n,1}(x)$ 最大. 因为这个原因, 我们也可以尝试 "Choice#2", 即在边界子区间 $[0, h]$ 和 $[1-h, 1]$ 的中点插入两个新的节点. 这时节点是

$$x_1 = 0,\ x_2 = \frac{1}{2}h,\ x_3 = h,\ x_4 = 2h, \cdots, x_{n+1} = (n-1)h,$$
$$x_{n+2} = 1 - \frac{1}{2}h,\ x_{n+3} = 1$$

利用spline得到一个三次样条函数, 把它记为 $s_{n,2}(x)$. 根据非节点插值条件, 节点 x_2 和节点 x_{n+2} 分别是 (4.73) 中的 z_1 和 z_2. 一般来说, $s_{n,2}(x)$ 是比 $s_{n,1}(x)$ 更精确的逼近.

对于区间 $[x_1, x_2]$ 和 $[x_2, x_3]$ 而言, 由spline产生的 $s_{n,2}(x)$ 是同一个三次多项式, 因此可以把 $[0, \frac{1}{2}h]$ 上的多项式用在整个区间 $[0, h]$ 上, 在区间 $[1-h, 1]$ 上也有类似的结论. 对于上述两种选择的插值结点, 在表 4-4 中给出最大误差

$$E_n^{(k)} = \max_{0 \leqslant x \leqslant 1} |f(x) - s_{n,k}(x)|$$

■

表 4-4 $f(x) = \mathrm{e}^x$ 的三次样条逼近

n	$E_n^{(1)}$	比率	$E_n^{(2)}$	比率
5	1.01E−4		1.11E−5	
10	6.92E−6	14.6	7.88E−7	14.1
20	4.56E−7	15.2	5.26E−8	15.0
40	2.92E−8	15.6	3.39E−9	15.5

习题

1. 考虑数据点 $\{(0,1),(1,1),(2,5)\}$.

(a) 求数据的分段线性插值函数;

(b) 求二次插值多项式;

(c) 求插值数据的自然三次样条.

对以上三种情况, 画出插值函数在区间 $0 \leqslant x \leqslant 2$ 上的图像.

2. 考虑数据

x	1	2	3	4	5
y	3	1	2	3	2

(a) 求分段线性插值函数 $l(x)$;

(b) 求插值数据且满足非节点边界条件 (4.73) 的三次样条函数 $s(x)$. 注意, 此时 $n = 3$, $x_1 = 1$, $x_2 = 3$, $x_3 = 5$, $z_1 = 2$, $z_2 = 4$.

画出区间 $1 \leqslant x \leqslant 5$ 上的 $s(x)$ 和 $l(x)$ 的图像.

3. 考虑数据

x	0	1/2	1	2	3
y	0	1/4	1	−1	−1

(a) 求数据的分段线性插值函数;

(b) 求分段二次插值函数;

(c) 求插值数据的自然三次样条;

(d) 求非节点插值的三次样条. 当使用 (4.73) 时, 令 $x_1 = 0$, $x_2 = 1$, $x_3 = 3$, 以及 $z_1 = \frac{1}{2}$, $z_2 = 2$.

画出所有四个函数在区间 $0 \leqslant x \leqslant 3$ 上的图像.

4. 对数据

x	0	1	2	2.5	3	4
y	1.4	0.6	1.0	0.65	0.6	1.0

重复习题 3 的工作. 但是注意分段二次插值需要加以修改. 使用 $[0,2]$ 和 $[2,3]$ 上的二次插值. 对于区间 $[3,4]$ 上的 $q(x)$, 构造 $[2.5,4]$ 上的二次插值多项式, 然后只把它使用在区间 $[3,4]$ 上. 6.4 节中给出的求解线性方程组的方法最适合用来求解三次样条函数的线性方程组.

5. 利用 MATLAB 的内置函数spline, 在区间 $-5 \leqslant x \leqslant 5$ 上对下列 x 的节点集插值函数 $f(x) = 1/(1+x^2)$.
   (i) $\{-5, -2.5, 0, 2.5, 5\}$.
   (ii) $\{-5, -3.5, -2, 0, 2, 3.5, 5\}$;
   (iii) $\{-5, -4.5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 4.5, 5\}$.
   对每一种情况, 画出样条函数和函数 $f(x)$ 的图像, 并与图 4-6 进行比较.
6. 证明: 边界条件 (4.71) 可分别导出方程

$$\begin{aligned}\frac{x_2-x_1}{3}M_1+\frac{x_2-x_1}{6}M_2&=\frac{y_2-y_1}{x_2-x_1}-f'(x_1)\\ \frac{x_n-x_{n-1}}{6}M_{n-1}+\frac{x_n-x_{n-1}}{3}M_n&=f'(x_n)-\frac{y_n-y_{n-1}}{x_n-x_{n-1}}\end{aligned} \tag{4.74}$$

   提示: 在 $[x_1, x_2]$ 和 $[x_{n-1}, x_n]$ 上求 (4.63) 的导数, 然后再利用 (4.71).
7. (a) 利用边界条件 (4.71),

$$s'(1) = -1, \quad s'(4) = -\frac{1}{16}$$

   求解插值数据 (4.66) 的三次样条.
   提示: 把习题 6 中的 (4.74) 与 (4.67) 相结合, 再求解由所得的四个方程所组成的线性方程组.
   (b) 对于函数 $f(x) = 1/x$ 以及由它生成的数据, 把自然样条 (4.68) 与上述三次样条进行比较. 对 $1 \leqslant x \leqslant 4$ 内的一些样本点计算 $f(x) - s(x)$.
8. (a) 利用边界条件 (4.72),

$$s''(1) = 2, \quad s''(4) = \frac{1}{32}$$

   求解插值数据 (4.66) 的三次样条.
   提示: 把 $M_1 = 2$ 和 $M_4 = \frac{1}{32}$ 与方程组 (4.67) 相结合.
   (b) 对于函数 $f(x) = 1/x$ 以及由它生成的数据, 把上述样条与自然三次样条 (4.68) 进行比较.
9. 求出满足下列条件的三次样条

$$s(0) = 0, s(1) = 1, s(2) = 2, s'(0) = 0, s''(2) = 2$$

   再画出它的图像.
10. 下列函数是区间 $0 \leqslant x \leqslant 2$ 上的一个三次样条吗?

$$s(x) = \begin{cases} (x-1)^3, & 0 \leqslant x \leqslant 1 \\ 2(x-1)^3, & 1 \leqslant x \leqslant 2 \end{cases}$$

11. 定义

$$s(x) = \begin{cases} -5+8x-6x^2+2x^3, & 1 \leqslant x \leqslant 2 \\ 27-40x+18x^2-2x^3, & 2 \leqslant x \leqslant 3 \end{cases}$$

   证明 $s(x)$ 是 $[1, 3]$ 上的三次样条函数. 试问它也是这个区间上的自然三次样条函数吗?

12. 定义

$$s(x)=\begin{cases}2x^3, & 0\leqslant x\leqslant 1\\ x^3+3x^2-3x+1, & 1\leqslant x\leqslant 2\\ 9x^2-15x+9, & 2\leqslant x\leqslant 3\end{cases}$$

证明 $s(x)$ 是 $[0,3]$ 上的三次样条函数. 试问它也是这个区间上的自然三次样条函数吗?

13. 定义

$$s(x)=\begin{cases}x^3-3x^2+2x+1, & 1\leqslant x\leqslant 2\\ -x^3+9x^2-22x+17, & 2\leqslant x\leqslant 3\end{cases}$$

试问 $s(x)$ 是 $[1,3]$ 上的三次样条函数吗? 它是这个区间上的自然三次样条函数吗?

14. 试问下列函数是 $[0,3]$ 上的一个三次样条吗?

$$s(x)=\begin{cases}x^3, & 0\leqslant x\leqslant 1\\ 2x-1, & 1\leqslant x\leqslant 2\\ 3x^2-9, & 2\leqslant x\leqslant 3\end{cases}$$

15. 定义

$$s(x)=\begin{cases}x^3+2x^2+1, & 1\leqslant x\leqslant 2\\ -2x^3+\beta x^2-36x+25, & 2\leqslant x\leqslant 3\end{cases}$$

对 β 的某一特定值, $s(x)$ 是 $[1,3]$ 上的三次样条函数. 求出 β 的这个值并证明 $s(x)$ 是 $[1,3]$ 上的三次样条函数. 试问它也是这个区间上的自然三次样条函数吗?

16. 是否存在系数 $\{a,b,c,d\}$ 的一种选择使得下列函数是三次样条:

$$s(x)=\begin{cases}(x+1)^3, & -2\leqslant x\leqslant -1\\ ax^3+bx^2+cx+d, & -1\leqslant x\leqslant 1\\ (x-1)^2, & 1\leqslant x\leqslant 2\end{cases}$$

17. (a) 公式 (4.63) 给出了区间 $x_{j-1}\leqslant x\leqslant x_j$ 上的 $s(x)$. 试给出 $x_j\leqslant x\leqslant x_{j+1}$ 上对应的公式.

(b) 对 $x_{j-1}\leqslant x\leqslant x_j$ 和 $x_j\leqslant x\leqslant x_{j+1}$, 推导 $s'(x)$ 的公式.

(c) 在 (b) 中得到的 $s'(x)$ 的两个公式中取 $x=x_j$, 并且要求它们相等. 导出公式 (4.64).

18. 设 $s(x)$ 是单个节点 c 上的三次样条, c 满足 $a<c<b$. 对于 $a\leqslant x\leqslant c$, 假设 $s(x)=0$. 证明存在某一常数 d, 使得

$$s(x)=d(x-c)^3,\quad c\leqslant x\leqslant b$$

19. (a) 给定区间 $[a,b]$ 并且设 $a<c<b$. 定义

$$\sigma_c(x)=\begin{cases}0, & a\leqslant x\leqslant c\\ (x-c)^3, & c\leqslant x\leqslant b\end{cases}$$

证明 $\sigma_c(x)$ 是 $[a,b]$ 上的三次样条函数.

(b) 设 $x_1 < x_2 < \cdots < x_n$ 并且 $p(x)$ 是次数 $\leqslant 3$ 的任一多项式. 对任意常数 $b_2, \cdots, b_{n-1}$, 定义

$$s(x) = \sum_{j=2}^{n-1} b_j \sigma_{x_j}(x) + p(x), \quad x_1 \leqslant x \leqslant x_n$$

定义中用到 (a) 中 $\sigma_c(x)$ 的定义. 证明 $s(x)$ 是 $[a,b] \equiv [x_1, x_n]$ 上的一个三次样条. 试问 $s'''(x)$ 在什么点不连续? 可以证明这个公式能够表示 $[x_1, x_n]$ 上的所有三次样条; 但是由于求解 $p(x)$ 和系数 $\{b_j\}_{j=2}^{n-1}$ 时呈病态, 所以我们不推荐使用它进行实际计算.

20. 为了研究三次样条插值的精度, 用一个程序包去构造 $0 \leqslant x \leqslant 1$ 上函数 $y = \mathrm{e}^x$ 的插值样条. 使用 $[0,1]$ 上的等距节点生成数据, 比如说用 10 个、20 个或者 40 个子区间. 在 4 倍于节点数的点上检验样条函数的精度. 对下面两种情况完成上述过程: (a) 三次自然插值的样条函数; (b) 满足 (4.71) 或者 (4.72) 的样条函数. 讨论你的结果. 试问当子区间个数翻倍的时候误差性态如何变化?

4.4 最佳逼近问题

本节我们考虑最佳可能逼近的概念. 我们用 $f(x) = \mathrm{e}^x$ 的泰勒多项式的一些改进来说明这一点.

为了理解可能改进泰勒多项式逼近的程度, 我们考虑最佳可能逼近的概念. 设 $f(x)$ 是某个区间 $a \leqslant x \leqslant b$ 上已知的连续函数. 若 $p(x)$ 是一个多项式, 则我们感兴趣的是度量区间 $[a,b]$ 上用 $p(x)$ 逼近 $f(x)$ 的最大可能的误差

$$E(p) = \max_{a \leqslant x \leqslant b} |f(x) - p(x)| \tag{4.75}$$

对每个次数 $n > 0$, 定义

$$\begin{aligned} \rho_n(f) &= \min_{\deg(p) \leqslant n} E(p) \\ &= \min_{\deg(p) \leqslant n} \left[\max_{a \leqslant x \leqslant b} |f(x) - p(x)| \right] \end{aligned} \tag{4.76}$$

这是用次数 $\leqslant n$ 的多项式可以达到的 $E(p)$ 的最小可能的值, 称它为**极小极大误差**(minimax error). 可以证明存在唯一的次数 $\leqslant n$ 的多项式, 它在 $[a,b]$ 上的极大误差是 $\rho_n(f)$. 称这个多项式为 n **阶极小极大多项式逼近**(minimax polynomial approximation), 把它记为 $m_n(x)$.

例 4.4.1 设在区间 $-1 \leqslant x \leqslant 1$ 上 $f(x) = \mathrm{e}^x$, 考虑 $f(x)$ 的线性多项式逼近. 由 1.2 节可知, 线性泰勒多项式是

$$t_1(x) = 1 + x \tag{4.77}$$

直接计算可得

$$\max_{-1\leqslant x\leqslant 1}|e^x - t_1(x)| \doteq 0.718 \tag{4.78}$$

利用这里未加讨论的一些方法, 得到区间 $[-1,1]$ 上 e^x 的线性极小极大多项式是

$$m_1(x) = 1.264\,3 + 1.175\,2x \tag{4.79}$$

并且

$$\max_{-1\leqslant x\leqslant 1}|e^x - m_1(x)| \doteq 0.279 \tag{4.80}$$

这两个线性多项式的图像与 e^x 的图像都由图 4-12 给出. ■

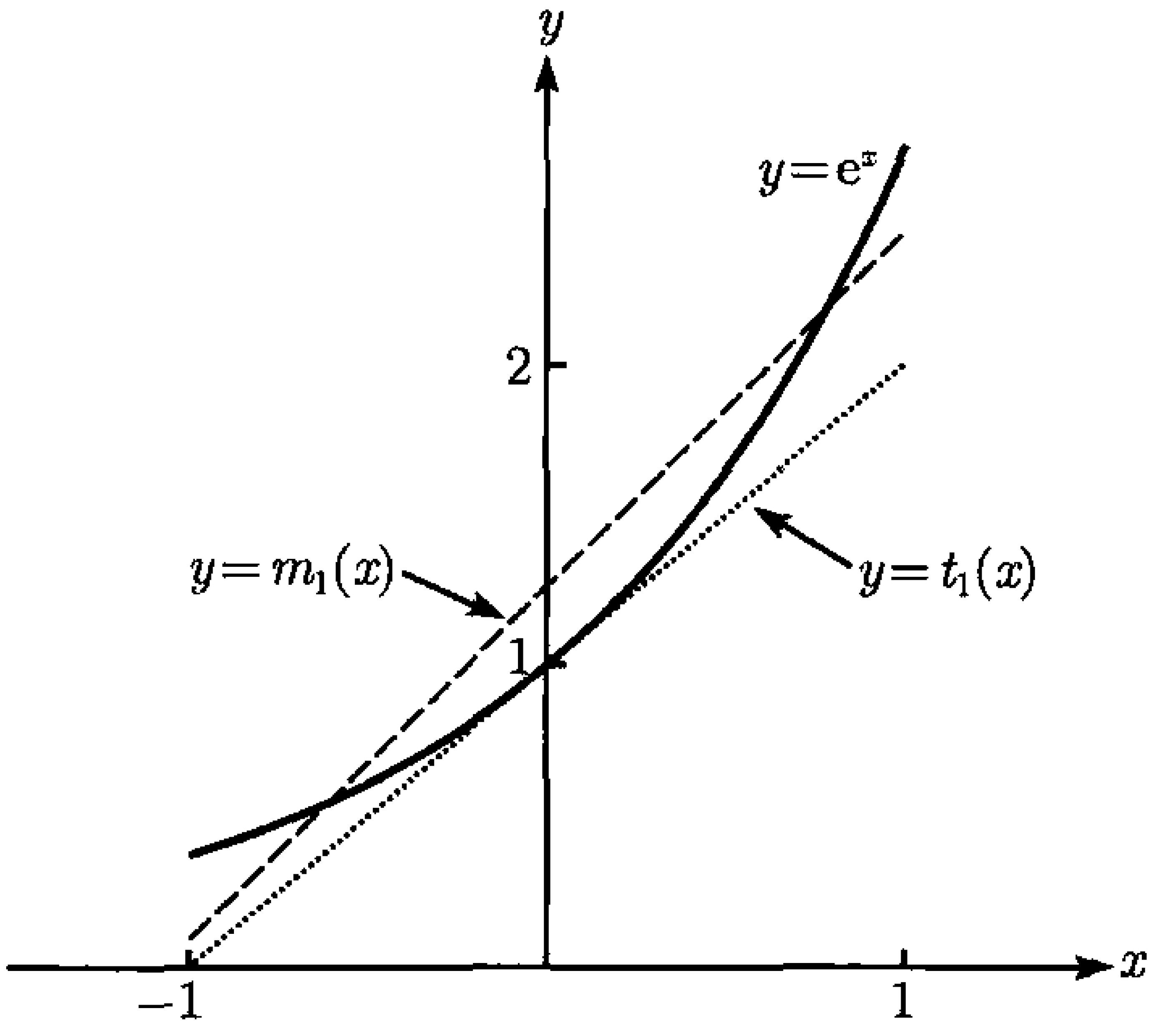


图 4-12　e^x 的两个线性逼近

例 4.4.2　仍然在 $-1 \leqslant x \leqslant 1$ 上假设 $f(x) = e^x$. 对 n 次泰勒多项式 $t_n(x)$ 和 n 次极小极大多项式逼近 $m_n(x)$, 比较它们的极大误差 (4.75). 对几个 n 的值, 其极大误差值由表 4-5 给出. 可以看出, 随着 n 的增加, 相对于 $t_n(x)$ 的精度, $m_n(x)$ 的精度变得越来越高. 这是极小极大逼近的一般特征. 由于 e^x 的泰勒级数收敛的非常快, 所以此例效果不如许多其他函数 $f(x)$ 那样显著.

表 4-5　$[-1,1]$ 上 e^x 的泰勒误差和极小极大误差

n	$t_n(x)$	极大误差 $m_n(x)$
1	7.18E−1	2.79E−1
2	2.18E−1	4.50E−2
3	5.16E−2	5.53E−3
4	9.95E−3	5.47E−4
5	1.62E−3	4.52E−5
6	2.26E−4	3.21E−6
7	2.79E−5	2.00E−7
8	3.06E−6	1.11E−8
9	3.01E−7	5.52E−10

为了进一步考察泰勒逼近与极小极大逼近之间的差异, 我们给出 $n=3$ 时误差的图像. 我们有

$$\begin{aligned} t_3(x) &= 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 \\ m_3(x) &= 0.994\,579 + 0.995\,668x + 0.542\,973x^2 + 0.179\,533x^3 \end{aligned} \tag{4.81}$$

误差的图像如图 4-13 和图 4-14 所示. 请注意两个图中纵坐标的尺度是不同的. ■

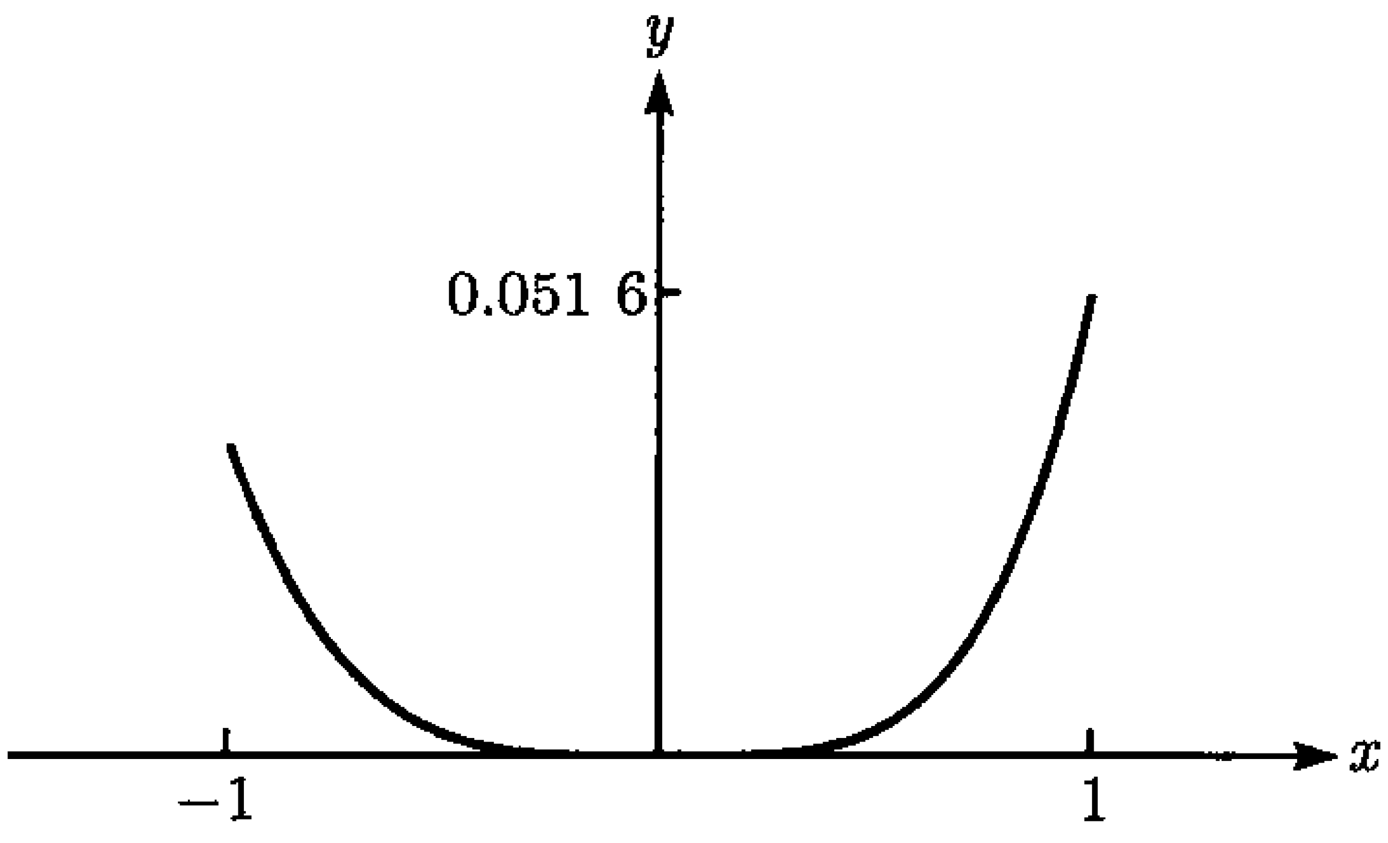


图 4-13　$e^x - t_3(x)$

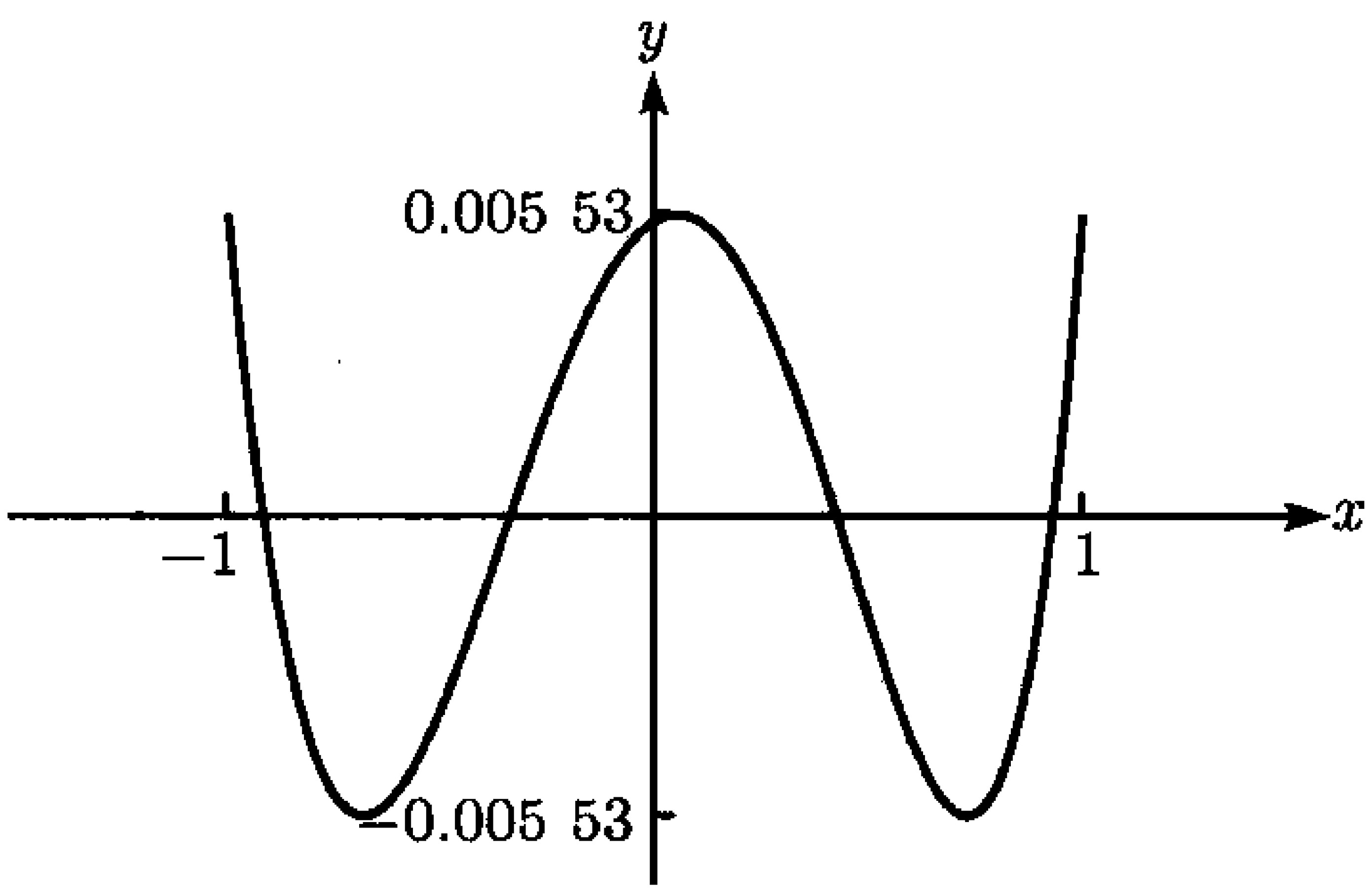


图 4-14　$e^x - m_3(x)$

这些例子说明了区间 $[a,b]$ 上逼近函数 $f(x)$ 的极小极大逼近 $m_n(x)$ 的几个一般性质. 首先, 就精度而言, $m_n(x)$ 通常是泰勒多项式 $t_n(x)$ 的一个显著的改进. 这意味着为了得到一个符合给定精度要求的逼近, 极小极大逼近多项式的次数通常远低于具有同等精度的泰勒多项式的次数. 正如泰勒多项式 $t_n(x)$ 的次数可能小于 n, $m_n(x)$ 的实际次数也可能小于 n, 但我们仍然称 $m_n(x)$ 为 "n 次极小极大逼近".

$m_n(x)$ 的第二个特征是误差 $f(x) - m_n(x)$ 的最大值分散在整个区间 $[a,b]$ 上. 相比之下, 泰勒误差 $f(x) - t_n(x)$ 在展开点附近的值远小于它在区间 $[a,b]$ 中其他点处的值. 从某种意义上说, 通过把误差值更加均匀地分配在整个区间上可以得到 $f(x) - m_n(x)$ 的较小的值. 第三个相伴的特征是误差 $f(x) - m_n(x)$ 在区间 $[a,b]$ 上振荡, 如图 4-14 所示. 可以证明, 这个误差在区间 $[a,b]$ 内改变符号至少 $n+1$ 次, 而且振荡的大小将会是相同的.

这些性质可以用于构造 $m_n(x)$, 不过鉴于问题的复杂性, $m_n(x)$ 的构造最好留给专家去完成. 在大多数计算机中心实验室都可以找到构造 $m_n(x)$ 的相关程序. 我们将用这些性质在 4.6 节中提出另一种方法, 这种方法会给出一个很接近极小极大逼近的多项式, 而且构造起来非常容易.

极小极大逼近的精度

从上面关于 $f(x) = \mathrm{e}^x$ 的例子中知道, 对相对较小的 n 来说, $m_n(x)$ 看上去就已经非常精确了. 对于像 e^x 和 $\cos(x)$ 等常见的函数而言, $m_n(x)$ 就更精确了. 假设 $f(x)$ 在区间 $[a, b]$ 上有无限阶连续导数, 并且设 $m_n(x)$ 是 $[a, b]$ 上 $f(x)$ 的 n 次极小极大逼近. 则极小极大误差满足

$$\rho_n(f) \leqslant \frac{[(b-a)/2]^{n+1}}{(n+1)!2^n} \max_{a \leqslant x \leqslant b} |f^{(n+1)}(x)| \tag{4.82}$$

由 4.6 节中的一些结果可以导出这个上界. 随着 n 的增大这个误差界不会总是变得更小, 但是对于很多常用函数 $f(x)$, 这个公式都能给出一个相当精确的界.

表 4-6　$\rho_n(\mathrm{e}^x)$ 的界

n	上界 (4.82)	$\rho_n(f)$
1	6.80E−1	2.79E−1
2	1.13E−1	4.50E−2
3	1.42E−2	5.53E−3
4	1.42E−3	5.47E−4
5	1.18E−4	4.52E−5
6	8.43E−6	3.21E−6
7	5.27E−7	2.00E−7

例 4.4.3　在 $-1 \leqslant x \leqslant 1$ 上设 $f(x) = \mathrm{e}^x$. 则 (4.82) 变为

$$\rho_n(\mathrm{e}^x) \leqslant \frac{\mathrm{e}}{(\mathrm{n}+1)!2^n} \tag{4.83}$$

对于不同的 n 值, 表 4-6 给出了这些上界的值, 还根据表 4-5 给出了 $\rho_n(f)$ 对应的精确值. 计算所得的值大约是真实值的 2.5 倍, 但这仍旧可以认为是 $\rho_n(f)$ 的一个相当精确的估计值. ■

习题

1. 证明 (4.78) 和 (4.80).
2. (a) 对于 $f(x) = \tan^{-1}(x)$, 给出泰勒逼近 $t_1(x)$ 和 $t_3(x)$. 并求出这两个逼近相对于 $\tan^{-1}(x)$ 在 $[-1, 1]$ 上的极大误差.

(b) 区间 $[-1,1]$ 上 $f(x)=\tan^{-1}(x)$ 的线性极小极大多项式和三次极小极大多项式分别是

$$m_1(x) = 0.833\ 278x$$
$$m_3(x) = 0.972\ 385\ 88x - 0.191\ 937\ 97x^3$$

求出 $[-1,1]$ 上它们的极大误差.

(c) 在 $[-1,1]$ 上画出 $f(x)-t_3(x)$ 和 $f(x)-m_3(x)$ 的图像.

3. 对于很多函数 $f(x)$ 来说, 可以利用各种恒等式缩减需要逼近的区间, 把一个很大的区间或者无限长度的区间变为一个相对较小的有限区间. 例如, 对于 $f(x)=\mathrm{e}^x$, 利用

$$\mathrm{e}^x = \frac{1}{\mathrm{e}^{-x}} \quad 如果\ x<0$$

而且对于某个 $m \geqslant 1$, 如果 $m \leqslant x < m+1$, 则有

$$\mathrm{e}^x = \mathrm{e}^m \mathrm{e}^y, \quad y = x-m, \quad 0 \leqslant y < 1$$

这两个恒等式把 e^x 的计算变为普通的乘法, 其中 $\mathrm{e}^m = \mathrm{e}\cdot\mathrm{e}\cdot\cdots\cdot\mathrm{e}$ 是 m 项 e 连乘, 而 e^y 的计算只需要在 0 与 1 之间的 y.

(a) 用上述方法, 计算区间 $[0,1]$ 上 $f(x)=\mathrm{e}^x$ 的界 (4.82) 并且令 $n=1,2,\cdots,7$. 把计算结果与表 4-6 中给出的结果进行比较.

(b) 为了检验 (a) 的精度, 计算区间 $[0,1]$ 上 $m_3(x)$ 的精确误差, 其中

$$m_3(x) = 0.999\ 455\ 2 + 1.016\ 602\ 3x + 0.421\ 703\ 0x^2 + 0.279\ 976\ 5x^3$$

画出 $[0,1]$ 上 $\mathrm{e}^x - m_3(x)$ 的图像.

4. 类似于习题 3 中区间缩减的讨论, 编写一段 MATLAB 代码, 把 $\cos(x)$ 的计算转化为 $\cos(y)$ 的计算, 其中与 x 相应的 y 值满足 $0 \leqslant y \leqslant \pi/2$.

5. (a) 对于 $n=1,2,\cdots,7$, 计算区间 $0 \leqslant x \leqslant \pi/2$ 上 $f(x)=\cos(x)$ 的界 (4.82).

(b) 为了检验 (a) 的精度, 计算区间 $[0,\pi/2]$ 上 $m_3(x) \approx \cos(x)$ 的精确误差, 其中

$$m_3(x) = 0.998\ 632\ 9 + 0.0296\ 140x - 0.600\ 861\ 6x^2 + 0.112\ 506\ 0x^3$$

(c) 画出 $[0,\pi/2]$ 上 $\cos(x)-m_3(x)$ 的图像.

6. 为什么在二进制计算机上逼近 $f(x)=\ln x$ 的自然区间是 $[a,b]=[1,2]$, 解释其原因. 给出一个算法, 把对一般的 $x>0$ 计算 $\ln x$ 转化为对 $[1,2]$ 中适当的 y 值计算 $\ln y$.

7. (a) 对 $f(x)=\ln x$, 在区间 $1 \leqslant x \leqslant 2$ 上界定极小极大逼近误差 $\rho_n(f)$. 对 $n \geqslant 1$ 的每一种情形找出一个上界, 并且使得当 $n \to \infty$ 时其上界序列收敛于零.

(b) 求 $f(x)=\ln x$ 关于 $x=\frac{3}{2}$ 处的三次泰勒多项式 $t_3(x)$. 在区间 $1 \leqslant x \leqslant 2$ 上它的最大误差是什么?

(c) 对于 $f(x)=\ln x$ 和区间 $[1,2]$,

$$m_3(x) = -1.492\ 776 + 2.112\ 632x - 0.729\ 104x^2 + 0.109\ 690x^3$$

求 $\rho_3(f)$ 并且画出 $[1,2]$ 上 $f(x)-m_3(x)$ 的图像. 比较 $t_3(x)$ 与 $m_3(x)$.

8. **(a)** 证明恒等式

$$\tan^{-1}(-x) = -\tan^{-1}(x)$$
$$\tan^{-1}(x) = \frac{\pi}{2} - \tan^{-1}\left(\frac{1}{x}\right), \quad x > 0$$

(b) 利用上述恒等式, 试问可用什么样的小区间来逼近下面的函数?

$$f(x) = \tan^{-1}(x), \quad -\infty < x < \infty$$

4.5 切比雪夫多项式

我们要引入一族多项式, 它们应用在数值分析的很多领域, 更一般地说, 是应用在数学和物理学中的*切比雪夫多项式*(Chebyshev polynomial). 本节中将给出这种多项式的一些性质, 4.6 节中将利用这些性质给出一个很接近极小极大逼近的多项式逼近.

对于整数 $n \geqslant 0$, 定义函数

$$T_n(x) = \cos(n \cos^{-1} x), \quad -1 \leqslant x \leqslant 1 \tag{4.84}$$

这个函数看上去不像是多项式, 但我们将证明它是一个 n 次多项式. 为了简化 (4.84) 的操作, 我们引入变量变换

$$\theta = \cos^{-1}(x) \text{ 或 } x = \cos(\theta), \quad 0 \leqslant \theta \leqslant \pi \tag{4.85}$$

则有

$$T_n(x) = \cos(n\theta) \tag{4.86}$$

切比雪夫多项式 $T_0(x), T_1(x), \cdots, T_4(x)$ 的图像如图 4-15 和图 4-16 所示.

例 4.5.1 (a) 对 $n = 0$, 有

$$T_0(x) = \cos(0 \cdot \theta) = 1$$

(b) 对 $n = 1$, 有

$$T_1(x) = \cos(\theta) = x$$

(c) 对 $n = 2$, 利用一个常用三角恒等式可得

$$T_2(x) = \cos(2\theta) = 2\cos^2(\theta) - 1 = 2x^2 - 1$$

■

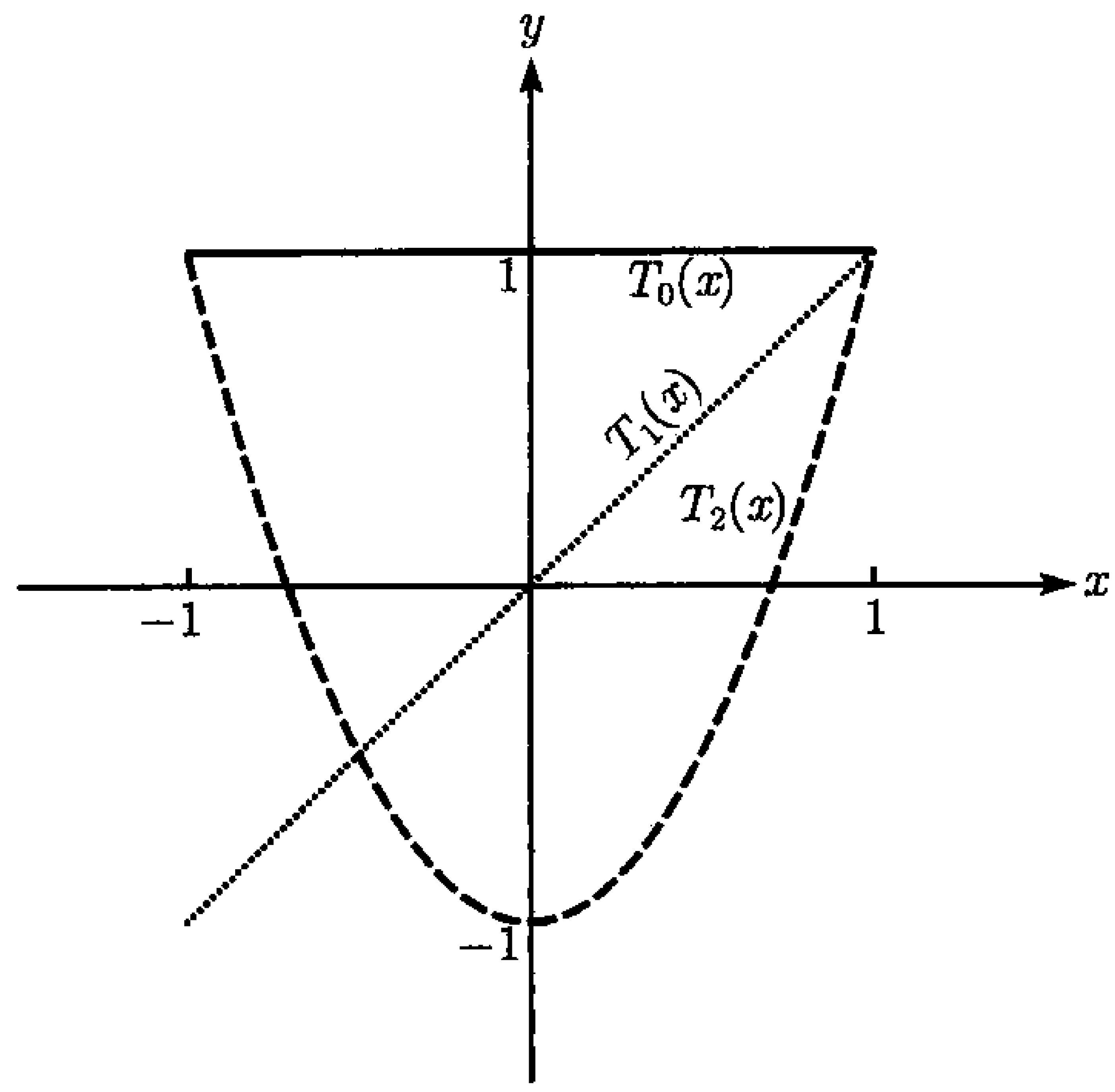


图 4-15　$T_0(x)$, $T_1(x)$ 和 $T_2(x)$ 的图像

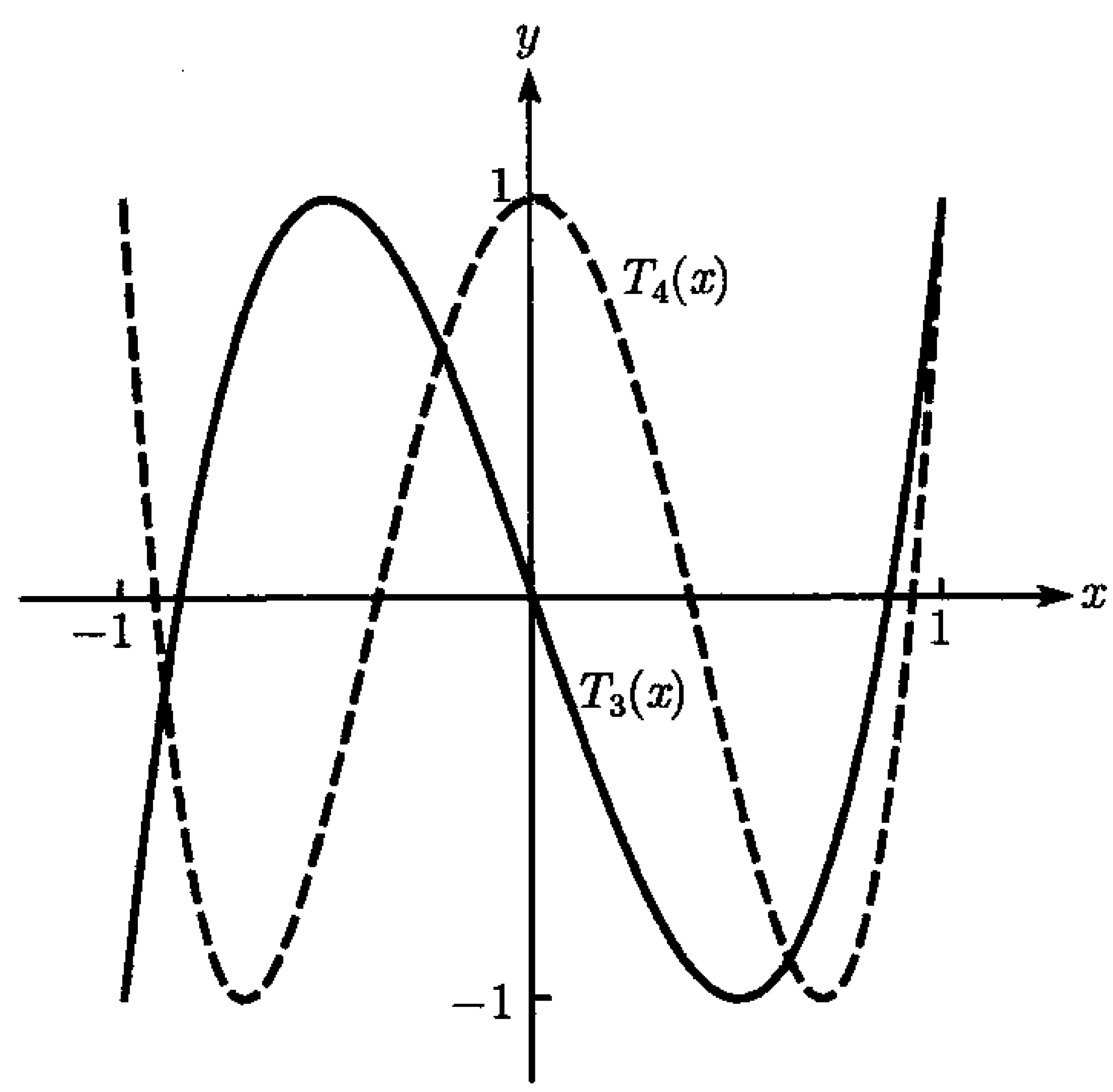


图 4-16　$T_3(x)$ 和 $T_4(x)$ 的图像

4.5.1　三项递推关系

回顾三角函数加法公式

$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$$

对于 $n \geqslant 1$, 利用这些恒等式可得

$$\begin{aligned}
T_{n+1}(x) &= \cos[(n+1)\theta] = \cos(n\theta + \theta) \\
&= \cos(n\theta)\cos(\theta) - \sin(n\theta)\sin(\theta) \\
T_{n-1}(x) &= \cos[(n-1)\theta] = \cos(n\theta - \theta) \\
&= \cos(n\theta)\cos(\theta) + \sin(n\theta)\sin(\theta)
\end{aligned}$$

两式相加, 然后利用 (4.85) 和 (4.86) 可得

$$T_{n+1}(x)+T_{n-1}(x)=2\ \cos(n\theta)\cos(\theta)=2xT_n(x)$$
$$T_{n+1}(x)=2xT_n(x)-T_{n-1}(x),\quad n\geqslant 1 \tag{4.87}$$

称它为切比雪夫多项式的**三项递推关系**(triple recursion relation). 在计算切比雪夫多项式的过程中, 通常使用这个递推关系而不是显式公式 (4.84).

例 4.5.2 (a) 在 (4.87) 中取 $n=2$. 利用已知的 $T_1(x)$ 和 $T_2(x)$ 的值, 可得

$$T_3(x)=2xT_2(x)-T_1(x)=2x(2x^2-1)-x=4x^3-3x$$

(b) 取 $n=3$, 则有

$$\begin{aligned}T_4(x)&=2xT_3(x)-T_2(x)=2x(4x^3-3x)-(2x^2-1)\\&=8x^4-8x^2+1\end{aligned}$$

■

4.5.2 最小取值范围性质

在介绍主要结果之前, 对所有 $n\geqslant 0$, 我们注意到

$$|T_n(x)|\leqslant 1,\quad -1\leqslant x\leqslant 1 \tag{4.88}$$

同样注意到

$$T_n(x)=2^{n-1}x^n+\text{低次项},\quad n\geqslant 1 \tag{4.89}$$

根据定义 (4.84), 利用 $|\cos(n\theta)|\leqslant 1$(参见 (4.86)) 可直接得到上述第一个结果. 用数学归纳法和递推公式 (4.87) 可证明第二个结果. 可以看出, 已经算出的 T_1, T_2, T_3 和 T_4 都符合 (4.89).

引入 $T_n(x)$ 的修正形式

$$\tilde{T}_n(x)=\frac{1}{2^{n-1}}T_n(x)=x^n+\text{低次项},\quad n\geqslant 1 \tag{4.90}$$

根据 (4.88) 和 (4.89) 得

$$|\tilde{T}_n(x)|\leqslant\frac{1}{2^{n-1}},\quad -1\leqslant x\leqslant 1,\quad n\geqslant 1 \tag{4.91}$$

我们把最高次项的系数是 1 的多项式称为**首一多项式**(monic polynomial). 公式 (4.91) 说明, 在区间 $-1\leqslant x\leqslant 1$ 上首一多项式 $\tilde{T}_n(x)$ 的取值范围为 $1/2^{n-1}$, 随着 n 的增大它会变得更小. 相比之下,

$$\max_{-1\leqslant x\leqslant 1}|x^n|=1$$

因此, x^n 是一个取值范围不随 n 的增加而变化的首一多项式.

定理 4.5.3　设整数 $n \geqslant 1$, 考虑所有可能的 n 次首一多项式. 则在 $[-1,1]$ 上具有最小极大绝对值的 n 次首一多项式是修正的切比雪夫多项式 $\tilde{T}_n(x)$, 它在 $[-1,1]$ 上的极大值是 $1/2^{n-1}$.

这个定理导出切比雪夫多项式若干有用的应用, 其中之一将在下节中介绍. 定理的证明留作习题 10.

习题

1. 求出 $T_5(x)$ 的显式多项式形式, 然后在 $-1 \leqslant x \leqslant 1$ 上画出它的图像.

2. 尽你所能说明 (4.89) 成立的理由.

3. 根据 (4.84), 我们知道在 $[-1,1]$ 上 $|T_n(x)| \leqslant 1$ 并且 $T_n(1)=1$. 求出关于点 x 的一般公式, 使得在点 x 处有 $T_n(x)=\pm 1$. 试问在区间 $[-1,1]$ 上有多少这样的点?
提示: 从一些特殊情形, 例如从 $n=3$ 开始讨论.

4. 对于 $2 \leqslant n \leqslant 10$, 计算 $T_n(0.5)$, 但不要使用定义 (4.84)~(4.86). 利用 $T_0(0.5)=1$ 和 $T_1(0.5)=0.5$.

5. 假设对于某个 x, 我们需要 $T_1(x), T_2(x), \cdots, T_n(x)$. 如果利用三项递推公式 (4.87) 产生这些值, 计算所需乘法的次数.

6. 设 $q(x)$ 是一个次数 $\leqslant n-1$ 的多项式, 考虑

$$\max_{-1 \leqslant x \leqslant 1} |x^n - q(x)|$$

它的最小可能值是多少? 求出相应于最小值的 $q(x)$.

7. 对于 $n, m \geqslant 0$ 并且 $n \neq m$, 证明

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} \mathrm{d}x = 0$$

称它为切比雪夫多项式的*正交性关系*(orthogonality relation).
提示: 利用 (4.84) 和变量替换 $x = \cos\theta$.

8. 函数

$$S_n(x) = \frac{1}{n+1} T'_{n+1}(x), \quad n \geqslant 0$$

是*第二类切比雪夫多项式*(Chebyshev polynomial of second kind).

(a) 给出 $S_0(x)$, $S_1(x)$, $S_2(x)$ 和 $S_3(x)$ 的表达式.

(b) 对于 $0 \leqslant \theta \leqslant \pi$, 证明

$$S_n(x) = \frac{\sin(n+1)\theta}{\sin\theta}, \quad \text{其中 } x = \cos\theta$$

(c) 利用 $\sin(\alpha \pm \beta)$ 的加法公式, 推导 $S_{n+1}(x)$ 的三项递推公式.

9. 单项式 $\{x^j | j \geqslant 0\}$ 可以表示为切比雪夫多项式的组合. 容易看出, $1 = T_0(x)$ 和 $x = T_1(x)$. 其次,

$$x^2 = \frac{1}{2}[T_2(x)+1] = \frac{1}{2}T_2(x) + \frac{1}{2}T_0(x)$$

(a) 类似地进行, 把 x^3 和 x^4 表示为切比雪夫多项式的组合.

(b) 证明: 对于每个 $n \geqslant 0$, 适当地选取系数 $\{a_{n,j}\}_{j=0}^{n}$, x^n 可以写成下列形式

$$x^n = a_{n,n}T_n(x) + a_{n,n-1}T_{n-1}(x) + \cdots + a_{n,0}T_0(x) \tag{4.92}$$

(c) 对于 $j > 0$, 证明

$$a_{n,0} = \frac{1}{\pi}\int_{-1}^{1} \frac{x^n}{\sqrt{1-x^2}}\mathrm{d}x \quad a_{n,j} = \frac{2}{\pi}\int_{-1}^{1} \frac{x^n T_j(x)}{\sqrt{1-x^2}}\mathrm{d}x$$

提示: 利用习题 7. 用 $T_j(x)/\sqrt{1-x^2}$ 乘以 (4.92) 的两端并且在 $[-1,1]$ 上积分.

10. 为了简要地说明定理 4.5.3 的证明, 考虑次数 $n=3$ 的情形.

(a) 利用习题 3, 求出使得

$$\tilde{T}_3(x) = \pm\frac{1}{4}$$

成立的 x 的值, 把它们记为 $x_3 < x_2 < x_1 < x_0$.

(b) 假设存在另一个首一多项式 $q(x)$ 使得

(i) $\max\limits_{-1\leqslant x\leqslant 1} |q(x)| < \frac{1}{2^{n-1}} = \frac{1}{4}$ (ii) $\deg(q) \leqslant 3$.

(c) 在 x_0, x_1, x_2, x_3 处计算多项式

$$R(x) = \tilde{T}_3(x) - q(x)$$

证明

$$R(x_0) > 0, \quad R(x_1) < 0, \quad R(x_2) > 0, \quad R(x_3) < 0$$

(d) 证明 $R(x)$ 的次数 $\leqslant 2$ 并且它至少有 3 个根.

(e) 与 (b) 中的假设 (i) 相反, 证明 $\tilde{T}_3(x) \equiv q(x)$. 这就证明不存在 3 次的首一多项式使得它在 $[-1,1]$ 上的最大绝对值小于 $\frac{1}{4}$.

4.6 近似极小极大逼近方法

因为我们在寻找已知函数 $f(x)$ 的多项式逼近, 考虑使用一个插值多项式看起来是合情合理的. 最直观的选择是选取相关区间 $a \leqslant x \leqslant b$ 上一组等距的插值结点. 但是很遗憾, 这样给出的插值多项式通常是 $f(x)$ 的一个很差的逼近, 其原因我们在这里不做探究. 这在 4.2 节中的图 4-6 和例 4.2.4 已经做了说明. 为了用更系统的方式考虑插值问题, 我们要用 4.2 节中的误差公式 (4.41) 对插值进行检查.

为了简化表达式, 我们选择特殊的区间 $-1 \leqslant x \leqslant 1$ 作为 $f(x)$ 的逼近区间, 而且一开始也把逼近多项式的次数限定为 $n=3$. 设 x_0, x_1, x_2, x_3 是 $[-1,1]$ 内的插值节点, $c_3(x)$ 表示在结点 x_0, x_1, x_2, x_3 处插值 $f(x)$ 的次数 $\leqslant 3$ 的多项式. 根据

(4.41), 对于 $-1 \leqslant x \leqslant 1$ 和 $[-1,1]$ 中的某个 c_x, 插值误差为

$$f(x) - c_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{4!} f^{(4)}(c_x) \tag{4.93}$$

我们选取节点 x_0, x_1, x_2, x_3 要使得区间 $[-1,1]$ 上 $|f(x) - c_x(x)|$ 的极大值尽可能地小.

观察 (4.93) 式右端, 可以看出, 我们能选择用来影响误差大小的量只有下列 4 次多项式

$$w(x) = (x-x_0)(x-x_1)(x-x_2)(x-x_3) \tag{4.94}$$

我们希望选取插值节点 x_0, x_1, x_2, x_3 使得

$$\max_{-1\leqslant x\leqslant 1} |w(x)| \tag{4.95}$$

尽可能地小.

如果展开 $w(x)$, 很容易得到

$$w(x) = x^4 + \text{低次项}$$

这是一个 4 次的首一多项式. 根据 4.5 节定理 4.5.3, (4.95) 可能的最小值由下式得到

$$w(x) = \frac{T_4(x)}{2^3} = \frac{1}{8}(8x^4 - 8x^2 + 1) \tag{4.96}$$

在这种情况下, (4.95) 的最小值是 $1/2^3$.

公式 (4.96) 隐含地定义了插值节点. 根据 (4.94), 节点是 $w(x)$ 的零点, 再由 (4.96) 知, 它们应当是 $T_4(x)$ 的零点. 可以通过对 (4.96) 右端的多项式数值求根来计算节点; 但是还有更简单的节点计算方法, 而且当多项式的次数变得更大时它仍然是一种非常有效的方法.

考虑定义 (4.84) 和 (4.85). 相应于 $n = 4$,

$$T_4(x) = \cos(4\theta), \quad x = \cos(\theta)$$

利用恒等式 $\cos(-\theta) = \cos(\theta)$, 我们知道当

$$\begin{aligned}
4\theta &= \pm\frac{\pi}{2}, \pm\frac{3\pi}{2}, \pm\frac{5\pi}{2}, \pm\frac{7\pi}{2}, \cdots \\
\theta &= \pm\frac{\pi}{8}, \pm\frac{3\pi}{8}, \pm\frac{5\pi}{8}, \pm\frac{7\pi}{8}, \cdots \\
x &= \cos\left(\frac{\pi}{8}\right), \cos\left(\frac{3\pi}{8}\right), \cos\left(\frac{5\pi}{8}\right), \cos\left(\frac{7\pi}{8}\right), \cos\left(\frac{9\pi}{8}\right), \cdots
\end{aligned} \tag{4.97}$$

时, 有 $T_4(x) = 0$. x 的前四个值是互不相同的, 但是后继的值不断重复前四个值. 因此, 由 (4.97) 我们得到节点的近似值为

$$\pm 0.382\ 683,\ \pm 0.923\ 880 \tag{4.98}$$

例 4.6.1 在 $[-1,1]$ 上设 $f(x) = \mathrm{e}^x$, 利用节点 (4.98) 给出三次插值多项式 $c_3(x)$. 表 4-7 中列举了节点、函数值以及根据插值多项式的牛顿差商公式 (4.35) 得到的差商. ■

表 4-7 $c_3(x)$ 的插值数据

i	x_i	$f(x_i)$	$f[x_0,\cdots,x_i]$
0	0.923 880	2.519 004 2	2.519 004 2
1	0.382 683	1.466 213 8	1.945 376 9
2	−0.382 683	0.682 028 8	0.704 742 0
3	−0.923 880	0.396 976 0	0.175 175 7

例 4.6.2 通过在大量的点上计算 $c_3(x)$, 我们得到

$$\max_{-1\leqslant x\leqslant 1} |\mathrm{e}^x - c_3(x)| \doteq 0.006\ 66 \tag{4.99}$$

图 4-17 给出了 $\mathrm{e}^x - c_3(x)$ 的图像. 把这个误差与表 4-5 中对应的值相比较, 得到

$$\rho_3(\mathrm{e}^x) \doteq 0.005\ 53$$

再把图 4-17 与 $m_3(x) \approx \mathrm{e}^x$ 的图 4-14 相比较. ■

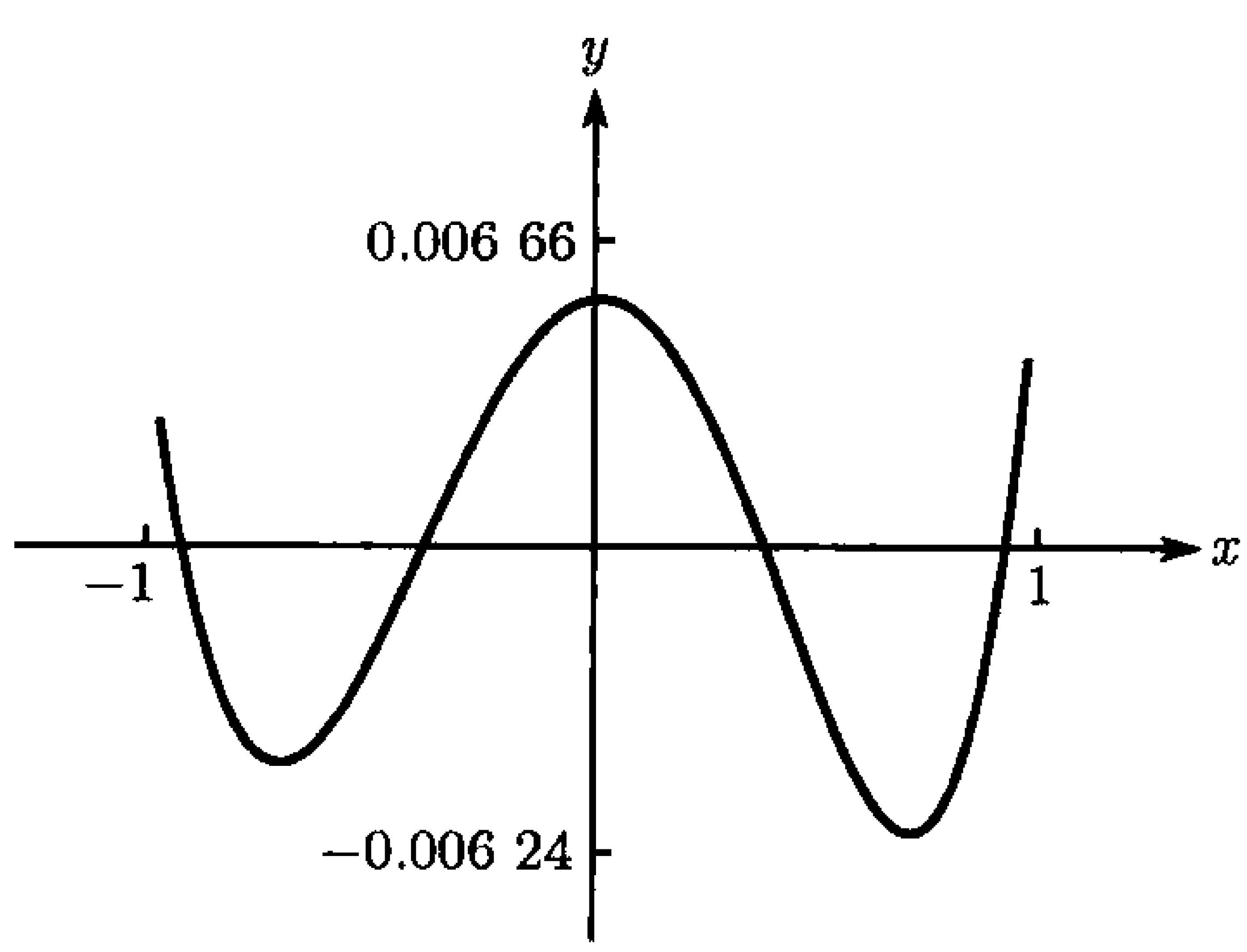


图 4-17 切比雪夫节点上的 $\mathrm{e}^x - c_3(x)$

把上述 $c_3(x)$ 的构造过程加以推广, 可得到区间 $[-1,1]$ 上 $f(x)$ 的 n 次近似极小极大逼近. 插值误差是

$$f(x)-c_n(x)=\frac{(x-x_0)\cdots(x-x_n)}{(n+1)!}f^{(n+1)}(c_x),\quad -1\leqslant x\leqslant 1 \tag{4.100}$$

我们设法极小化

$$\max_{-1\leqslant x\leqslant 1}|(x-x_0)\cdots(x-x_n)| \tag{4.101}$$

这个被极小化的多项式是 $n+1$ 次的首一多项式. 由定理 4.5.3 知, 可由首一多项式

$$\frac{1}{2^n}T_{n+1}(x)$$

达到最小值. 因此, 插值节点是 $T_{n+1}(x)$ 的零点. 根据 (4.97) 的推导过程, 得到节点如下

$$x_j=\cos\left(\frac{2j+1}{2n+2}\pi\right),\quad j=0,1,\cdots,n \tag{4.102}$$

在区间 $[-1,1]$ 上这 $n+1$ 节点处插值 $f(x)$ 就得到 n 次的近似极小极大逼近 $c_n(x)$.

例 4.6.3 设 $f(x)=\mathrm{e}^x$. 表 4-8 中给出了 $c_n(x)$ 在 $[-1,1]$ 上的极大误差. 为了做比较, 表中也给出了对应的极小极大误差 $\rho_n(x)$. 这些统计数字表明, 在实际问题中 $c_n(x)$ 是极小极大逼近 $m_n(x)$ 的一个令人满意的替代品. ■

表 4-8 $[-1,1]$ 上 e^x 的近似极小极大误差

n	$\max\|\mathrm{e}^x-c_n(x)\|$	$\rho_n(\mathrm{e}^x)$
1	3.72E−1	2.79E−1
2	5.65E−2	4.50E−2
3	6.66E−3	5.53E−3
4	6.40E−4	5.47E−4
5	5.18E−5	4.52E−5
6	3.62E−6	3.21E−6

MATLAB 程序 我们给出一个程序用来构造和计算区间 $[-1,1]$ 上已知函数 $f(x)$ 的近似极小极大逼近 $c_n(x)$. 多项式 $c_n(x)$ 表述为插值多项式的牛顿差商形式, 可利用 4.1 节 (4.39) 中嵌套相乘的方法来计算多项式的值. 注意在程序中使用了 4.1 节中的程序`divdif`和`interp`.

程序中被逼近的函数是 $f(x)=\mathrm{e}^x$, 但是利用函数子程序fcn很容易把它替换为某个其他的函数. 通过计算区间 $[-1,1]$ 内 501 个等距节点上的误差来估计最大误差

$$\max_{-1\leqslant x\leqslant 1}|\mathrm{e}^x-c_n(x)|$$

程序输出插值节点、对应的函数值和差商, 因此可以用另一个程序来计算 $c_n(x)$.

```
function [nodes, fcn_values, div_diff_fcn] = chebyshev_interp(n)
%
% This creates an interpolant of degree n to the function
% fcn(x) on [-1,1], which is given below by a function
% subprogram.  The nodes are the Chebyshev zeroes of the
% degree n+1 Chebyshev polynomial on [-1,1].  The program
% gives two plots:  first the true function and its
% interpolant, and second, the error in the interpolation.

% Create the nodes and associated divided differences.
h = pi/(2*(n+1));
nodes = cos(h*[1:2:2*n+1]);
fcn_values = fcn(nodes);
div_diff_fcn = divdif(nodes,fcn_values);

% Create the points at which the functions are to be
% graphed.
x_eval = -1:.002:1;
true_fcn = fcn(x_eval);
y_eval = interp(nodes,div_diff_fcn,x_eval);

% Create the window for the graph of the function
% and its interpolant.
m = min([min(true_fcn),min(y_eval)]);
M = max([max(true_fcn),max(y_eval)]);
axis([-1.1,1.1,m,M])
hold on

% Create the graph of the function and its interpolant.
plot(x_eval,true_fcn,'r')
plot(x_eval,y_eval,':')
legend('True function','Interpolant',0)
plot(nodes,fcn_values,'.','MarkerSize',6)
hold off
pause
clf

% Create the window for the graph of the error.
```

```
error = true_fcn - y_eval;
M = max(error);
m = min(error);
axis([-1.1,1.1,m,M])
hold on

% Create the graph of the error in the interpolant.
plot(x_eval,error,'r')
hold off

% Print the maximum error.
disp(['maximum error = ',num2str(max(abs(error)))])

function fval = fcn(x)
fval = exp(x);
```

对于一个实际问题，我们很可能会使用某种精确的但是效率低的方法，通常利用泰勒多项式来求函数的值. 这个逼近应该在函数子程序`fcn(x)`中给出，而主程序使用这个子程序产生一个更有效的近似. 另一个问题是，对于大多数函数而言，$[-1,1]$ 并不一定是这些函数被逼近的区间. 这种限制在习题 4 中得以消除.

奇函数和偶函数

关于这个算法的使用有必要提醒一下读者. 如果 $f(x)$ 是 $[-1,1]$ 上的所谓偶函数或者奇函数，则 n 的选取会受些限制. 如果对所有 x，都有

$$f(-x)=f(x) \tag{4.103}$$

我们称函数 $f(x)$ 是偶(even) 的. 例如 $f(x)=\cos(x)$. 这种函数的图像关于 y 轴对称. 如果对所有 x，都有

$$f(-x)=-f(x) \tag{4.104}$$

我们称函数 $f(x)$ 是奇(odd) 的. 例如 $f(x)=\sin(x)$. 这种函数关于原点对称.

对于这两种情况，如下选取上述算法中的 n：

$$\begin{aligned}&\text{如果}f(x)\text{是奇的，则选择}n\text{是偶数}\\&\text{如果}f(x)\text{是偶的，则选择}n\text{是奇数}\end{aligned} \tag{4.105}$$

这样得到的 $c_n(x)$ 的次数只有 $n-1$，但会给出一个适当的公式. 如果一个多项式是 (4.104) 中所示的奇函数，则 $c_n(x)$ 只含奇次项；而如果 $c_n(x)$ 是偶函数，则它只有偶次项. 限制 (4.105) 将在程序使用过程中考虑到这个问题，同时也给出良好的精度. 进一步的讨论参见 Atkinson(1989, 4.7 节).

习题

1. (a) 证明在 (4.97) 中只有四个不同的值.

   (b) 证明 $T_{n+1}(x)$ 的零点由 (4.102) 给出.

2. 对于区间 $[-1,1]$, 给出本节中线性的近似极小极大逼近的插值节点. 给出区间 $[-1,1]$ 上 $f(x)=\mathrm{e}^x$ 的线性的近似极小极大逼近.

3. (a) 执行本节中给出的程序. 包括差商和节点 (或者误差) 的输出, 这些误差可以用来产生误差函数的图像.

   (b) 在区间 $[-1,1]$ 上计算 $f(x)=\tan^{-1}(x)$ 的三次近似极小极大逼近. 并把它与 4.4 节习题 2(b) 中给出的三次极小极大逼近进行比较.

   注: 因为 $\tan^{-1}(x)$ 在 $[-1,1]$ 上是奇函数, 基于 (4.105), 在程序中使用 $n=4$.

4. 对于大多数函数而言, $[-1,1]$ 并不是被逼近函数的区间. 假设要在区间 $a\leqslant x\leqslant b$ 上计算 $g(t)$. 在 $[-1,1]$ 上定义一个新函数 $f(x)$ 如下

$$f(x)=g\left[\frac{(b+a)+x(b-a)}{2}\right],\quad -1\leqslant x\leqslant 1$$

   这里

$$t=\frac{1}{2}\left[(b+a)+x(b-a)\right]$$

   是一个线性变量替换. 我们现在可在 $[-1,1]$ 上逼近 $f(x)$.

   作为一个特例, 对于 $0\leqslant t\leqslant 1$, 给出 $g(t)=\mathrm{e}^t$ 的三次近似极小极大逼近. 把它与 4.4 节习题 3(b) 中给出的极小极大逼近进行比较.

5. 对于 $0\leqslant t\leqslant \pi/2$, 求出 $g(t)=\cos(t)$ 的三次近似极小极大逼近. 把它与 4.4 节习题 5(b) 中给出的极小极大逼近进行比较.

6. 对 $0\leqslant t\leqslant 1$ 求出 $g(t)=\tan^{-1}(t)$ 的近似极小极大逼近的极大误差. 对于 $n=1,2,\cdots,6$ 做上述工作.

7. 对于 $g(t)=\ln(t)$, $1\leqslant t\leqslant 2$, 重复习题 6 中的工作.

8. 对于

$$g(t)=\frac{1}{t}\int_0^t \frac{\sin(x)}{x}\mathrm{d}x,\quad -1\leqslant t\leqslant 1$$

   重复习题 6 中的工作. 为了最初计算 $g(t)$, 利用区间 $-1\leqslant t\leqslant 1$ 上精度为 5×10^{-15} 的泰勒多项式逼近. 仿效 1.3 节中函数 Sint x 的例子编写一个 MATLAB 函数计算这个多项式.

9. 对于函数

$$g(t)=\frac{1}{t}\int_0^t \frac{\mathrm{e}^u-1}{u}\mathrm{d}u,\quad -1\leqslant t\leqslant 1$$

   重复习题 8 中的工作.

10. 证明: 如果一个多项式 $p(x)$ 是满足 (4.103) 的偶函数, 则 $p(x)$ 中的所有奇次项都是零. 把这个结果扩展到多项式是奇函数的情形.

11. 利用 (4.100) 和 (4.101), 结合定理 4.5.3, 在 $[a,b]=[-1,1]$ 的情况下推导出 4.4 节中的界 (4.82).

4.7 最小二乘逼近

上节中利用适当选取的节点上的多项式插值, 我们给出了近似极小极大逼近多项式. 另一种方法是在逼近区间上寻找一个具有小的“平均误差”的逼近. 如果在区间 $a \leqslant x \leqslant b$ 上, 多项式 $p(x)$ 逼近函数 $f(x)$, 则逼近的平均误差的一个方便的定义如下

$$E(p;f) \equiv \sqrt{\frac{1}{b-a}\int_a^b [f(x)-p(x)]^2 \mathrm{d}x} \tag{4.106}$$

它也称为用 $p(x)$ 逼近 $f(x)$ 的**均方根误差**(root-mean-square-error). 我们先在相对简单的情况下说明它的用途. 首先注意, 对于不同选取的 $p(x)$ 极小化 $E(p;f)$ 等价于极小化

$$\int_a^b [f(x)-p(x)]^2 \mathrm{d}x \tag{4.107}$$

因而, 我们可以省去平方根以及相乘的因子 (虽然极小值一般是不同的).

例 4.7.1 设 $f(x) = \mathrm{e}^x$ 并且 $p(x) = \alpha_0 + \alpha_1 x$, 其中 α_0 和 α_1 任意取值. 我们要在区间 $[-1,1]$ 上逼近 $f(x)$. 我们选择 α_0 和 α_1 使积分

$$g(\alpha_0,\alpha_1) \equiv \int_{-1}^1 [\mathrm{e}^x - \alpha_0 - \alpha_1 x]^2 \mathrm{d}x \tag{4.108}$$

极小化. 展开上式, 我们得到

$$g(\alpha_0,\alpha_1) = \int_{-1}^1 \{\mathrm{e}^{2x} + \alpha_0^2 + \alpha_1^2 x^2 - 2\alpha_0 \mathrm{e}^x - 2\alpha_1 x \mathrm{e}^x + 2\alpha_0\alpha_1 x\}\mathrm{d}x \tag{4.109}$$

积分后得到如下形式的表达式

$$g(\alpha_0,\alpha_1) = c_1\alpha_0^2 + c_2\alpha_1^2 + c_3\alpha_0\alpha_1 + c_4\alpha_0 + c_5\alpha_1 + c_6$$

其中的系数 $\{c_1,\cdots,c_6\}$ 可由积分算得. 这是两个变量 α_0 和 α_1 的二次多项式. 它的极小值可通过求解下列联立方程组得到

$$\frac{\partial g}{\partial \alpha_0} = 0, \quad \frac{\partial g}{\partial \alpha_1} = 0 \tag{4.110}$$

上式中我们使用了 g 关于 α_0 和 α_1 的导数. 在这种情况下直接对 (4.108) 求导数更为简单, 我们得到

$$2\int_{-1}^1 [\mathrm{e}^x - \alpha_0 - \alpha_1 x](-1)\mathrm{d}x = 0$$

$$2\int_{-1}^1 [\mathrm{e}^x - \alpha_0 - \alpha_1 x](-x)\mathrm{d}x = 0$$

化简得到

$$2\alpha_0 = \int_{-1}^{1} \mathrm{e}^x \mathrm{d}x = \mathrm{e} - \mathrm{e}^{-1}$$

$$\frac{2}{3}\alpha_1 = \int_{-1}^{1} x\mathrm{e}^x \mathrm{d}x = 2\mathrm{e}^{-1}$$

$$\alpha_0 = \frac{\mathrm{e} - \mathrm{e}^{-1}}{2} \doteq 1.175\,2$$

$$\alpha_1 = 3\mathrm{e}^{-1} \doteq 1.103\,6$$

利用这些 α_0 和 α_1 的值, 我们得到线性逼近:

$$l_1(x) = \alpha_0 + \alpha_1 x$$

称之为 e^x 在最小二乘意义上的最佳线性逼近. 其误差为

$$\max_{-1\leqslant x\leqslant 1} |\mathrm{e}^x - l_1(x)| \doteq 0.439$$

它的图像由图 4-18 给出. 我们对比本章中曾经学习过的 e^x 的线性逼近的四种形式, 每种形式的极大误差和均方根误差由表 4-9 给出. ■

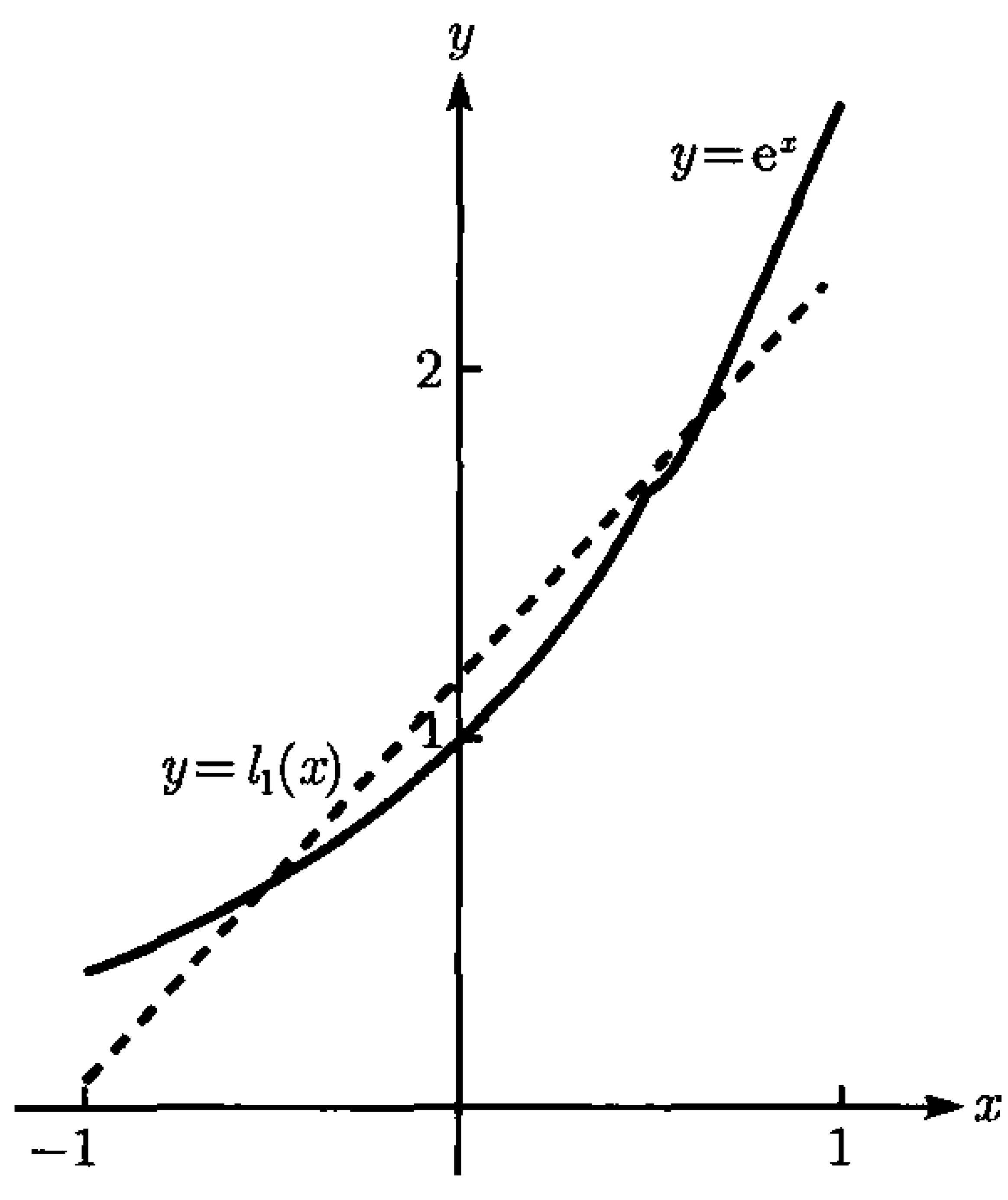


图 4-18 e^x 的线性最小二乘逼近

表 4-9 e^x 的线性逼近的误差

逼近	极大误差	均方根误差
泰勒 $t_1(x)$	0.718	0.246
最小二乘 $\ell_1(x)$	0.439	0.162
切比雪夫 $c_1(x)$	0.372	0.184
极小极大 $m_1(x)$	0.279	0.190

给定整数 $n \geqslant 0$, 我们考虑 (4.106) 中对于一般区间 $[a,b]$ 上一般函数 $f(x)$ 的 $E(p;f)$ 的极小化问题. 我们要寻找一个次数小于或等于 n 的多项式 $p(x)$, 它使 (4.107) 达到极小. 模仿前面的例题, 记

$$p(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_n x^n \tag{4.111}$$

我们定义

$$g(\alpha_0, \alpha_1, \cdots, \alpha_n) \equiv \int_{-1}^{1} [f(x) - \alpha_0 - \alpha_1 x - \cdots - \alpha_n x^n]^2 \mathrm{d}x \tag{4.112}$$

并寻找极小化这个积分的系数 $\alpha_0, \alpha_1, \cdots, \alpha_n$. 如果展开积分, 可以证明$g(\alpha_0, \alpha_1, \cdots, \alpha_n)$ 是关于 $n+1$ 个变量 $\alpha_0, \alpha_1, \cdots, \alpha_n$ 的一个二次多项式.

借助于下列条件

$$\frac{\partial g}{\partial \alpha_i} = 0, \quad i = 0, 1, \cdots, n$$

可以求出 $g(\alpha_0, \alpha_1, \cdots, \alpha_n)$ 的一个极小化解, 使 g 达到极小的一组元素 $\alpha_0, \alpha_1, \cdots, \alpha_n$ 必须满足这 $n+1$ 个方程. 处理这些条件导致一个联立线性方程组. 为了更好地理解线性方程组的形式, 考虑 $[a,b] = [0,1]$ 的特殊情况. 这时, 线性方程组是

$$\sum_{j=0}^{n} \frac{\alpha_j}{i+j+1} = \int_0^1 x^i f(x) \mathrm{d}x, \quad i = 0, 1, \cdots, n \tag{4.113}$$

我们将在第 6 章学习联立线性方程组的解法. 在 6.5 节中我们会看到这个线性方程组是 "病态" 的, 很难精确地得到其解, 即便是对 $n = 5$ 这样适度大小的 n 来说也是这样. 所以, 这不是一种求 (4.107) 中 $E(p;f)$ 的极小化解的好方法.

4.7.1　勒让德多项式

极小化 $E(p;f)$ 的一种更好的方法需要引入一组特殊多项式, 即**勒让德多项式**(Legendre polynomial). 它们的定义如下

$$\begin{aligned} P_0(x) &= 1 \\ P_n(x) &= \frac{1}{n!2^n} \cdot \frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2-1)^n], \quad n = 1, 2, \cdots \end{aligned} \tag{4.114}$$

例如

$$\begin{aligned} P_1(x) &= x \\ P_2(x) &= \frac{1}{2}(3x^2 - 1) \\ P_3(x) &= \frac{1}{2}(5x^3 - 3x) \\ P_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3) \end{aligned} \tag{4.115}$$

它们的图像如图 4-19 所示. 正如我们将在后面看到的那样, 当一个多项式用勒让德多项式来表示时, 可以直接确定最小二乘逼近的表示式的系数.

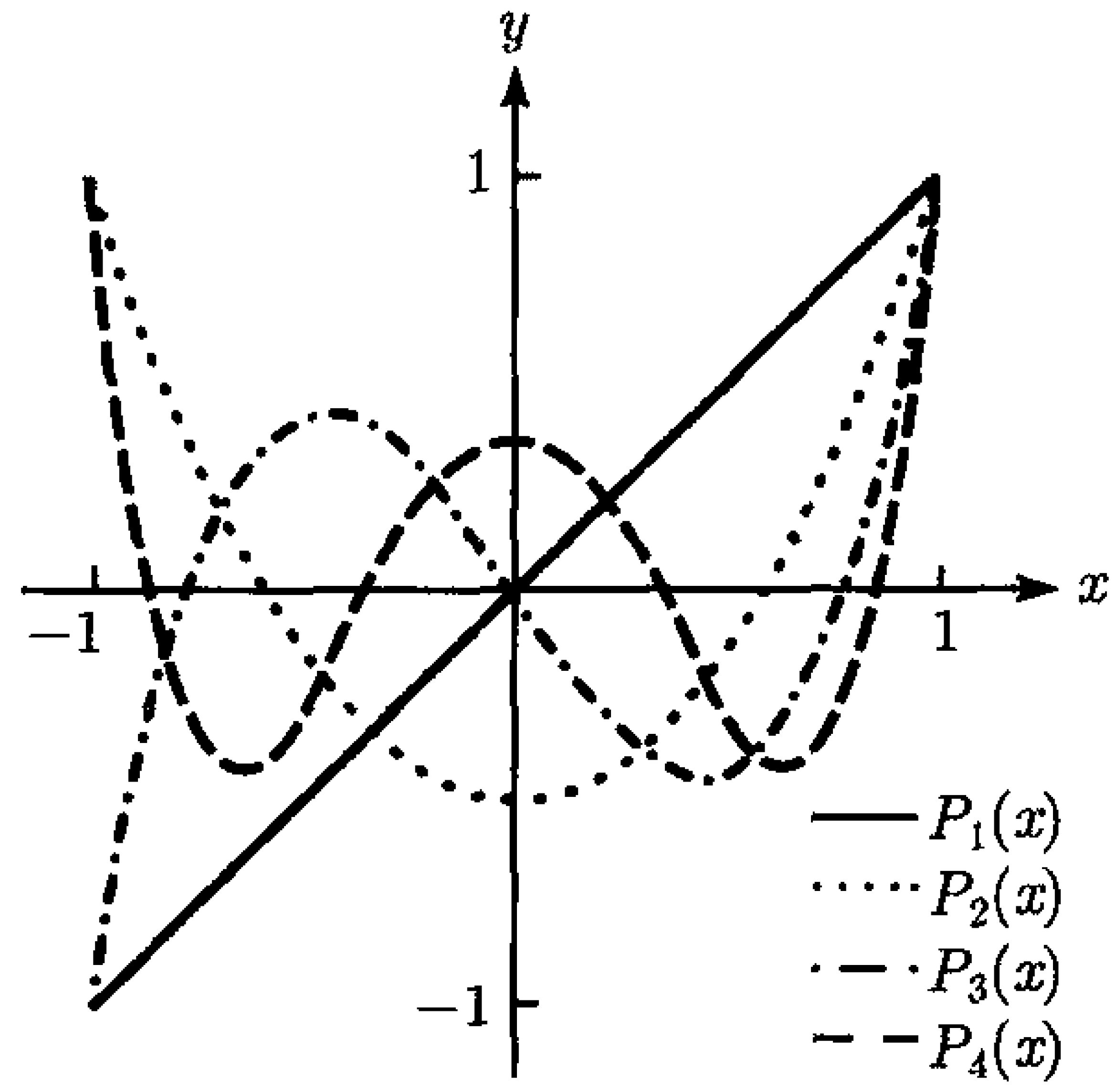


图 4-19 1 次、2 次、3 次和 4 次勒让德多项式

勒让德多项式有许多特殊性质, 而且在数值分析和应用数学中有广泛的应用. 我们给出它们的若干性质, 其证明过程参见 Atkinson(1989, 第 4 章). 对一般的函数 $f(x)$ 和 $g(x)$, 我们引进记号

$$(f,g)=\int_a^b f(x)g(x)\mathrm{d}x \tag{4.116}$$

- 次数和规范化:

$$\deg P_n=n,\quad P_n(1)=1,\quad n\geqslant 0$$

- 三项递推关系:

$$P_{n+1}(x)=\frac{2n+1}{n+1}xP_n(x)-\frac{n}{n+1}P_{n-1}(x),\quad n\geqslant 1 \tag{4.117}$$

- 正交性和大小:

$$(P_i,P_j)=\begin{cases}0, & i\neq j\\ \dfrac{2}{2j+1}, & i=j\end{cases} \tag{4.118}$$

- 零点:

$$\begin{gathered}P_n(x)\text{的所有零点位于区间}-1<x<1\text{中},\\ \text{而且,所有零点都是}P_n(x)\text{的单根}\end{gathered} \tag{4.119}$$

- 基底: 每个次数 $\leqslant n$ 的多项式 $p(x)$ 都可表示为下列形式

$$p(x)=\sum_{j=0}^{n}\beta_j P_j(x) \tag{4.120}$$

其中 $\beta_0,\beta_1,\cdots,\beta_n$ 的选取由 $p(x)$ 唯一确定.

4.7.2 求解最小二乘逼近

我们只考虑在区间 $[-1,1]$ 上求解最小二乘逼近问题. 正如 4.6 节习题 4 中讨论的那样, 其他区间 $[a,b]$ 上的逼近问题可利用变量的线性替换来完成. 利用 (4.116) 中的定义, 我们要极小化的量 (4.107) 可写为

$$(f-p,f-p) \tag{4.121}$$

首先, 我们把 $p(x)$ 写为 (4.120) 的形式, 再把它代入 (4.121), 得到

$$\tilde{g}(\beta_0,\beta_1,\cdots,\beta_n)\equiv(f-p,f-p)=\left(f-\sum_{j=0}^{n}\beta_jP_j,\ f-\sum_{i=0}^{n}\beta_iP_i\right)$$

利用性质 (4.118), 可以把上式展开成:

$$\tilde{g}=(f,f)-\sum_{j=0}^{n}\frac{(f,P_j)^2}{(P_j,P_j)}+\sum_{j=0}^{n}(P_j,P_j)\left[\beta_j-\frac{(f,P_j)}{(P_j,P_j)}\right]^2$$

仔细考察上式, 可以看出, 当

$$\beta_j=\frac{(f,P_j)}{(P_j,P_j)},\quad j=0,1,\cdots,n \tag{4.122}$$

时, $\tilde{g}$ 是最小的. 并且对于这样选取的系数, 极小值是

$$\tilde{g}=(f,f)-\sum_{j=0}^{n}\frac{(f,P_j)^2}{(P_j,P_j)}$$

我们称

$$l_n(x)=\sum_{j=0}^{n}\frac{(f,P_j)}{(P_j,P_j)}P_j(x) \tag{4.123}$$

为 $[-1,1]$ 上 $f(x)$ 的n *次最小二乘逼近*(least square approximation of degree n).

例 4.7.2 我们继续例 4.7.1 的讨论, 其中 $f(x)=\mathrm{e}^x$ 定义在 $[-1,1]$ 上. 我们使用 $n=3$ 时的公式 (4.123). 表 4-10 给出了系数 $\{\beta_0,\beta_1,\beta_2,\beta_3\}$.

结合 (4.115) 中给出的公式, 我们得到

$$l_3(x)=0.996\ 294+0.997\ 955x+0.536\ 722x^2+0.176\ 139x^3$$

$\mathrm{e}^x-l_3(x)$ 的图像如图 4-20 所示, 并且可以看出

$$\max_{-1\leqslant x\leqslant 1}|\mathrm{e}^x-l_3(x)|\doteq 0.011\ 2$$

我们比较本章中学过的 e^x 的三次逼近的四种形式, 每种形式的极大误差和均方根误差在表 4-11 中给出. ■

表 4-10 e^x 的三次最小二乘逼近的系数 β_j

j	0	1	2	3
β_j	1.175 20	1.103 64	0.357 81	0.070 46

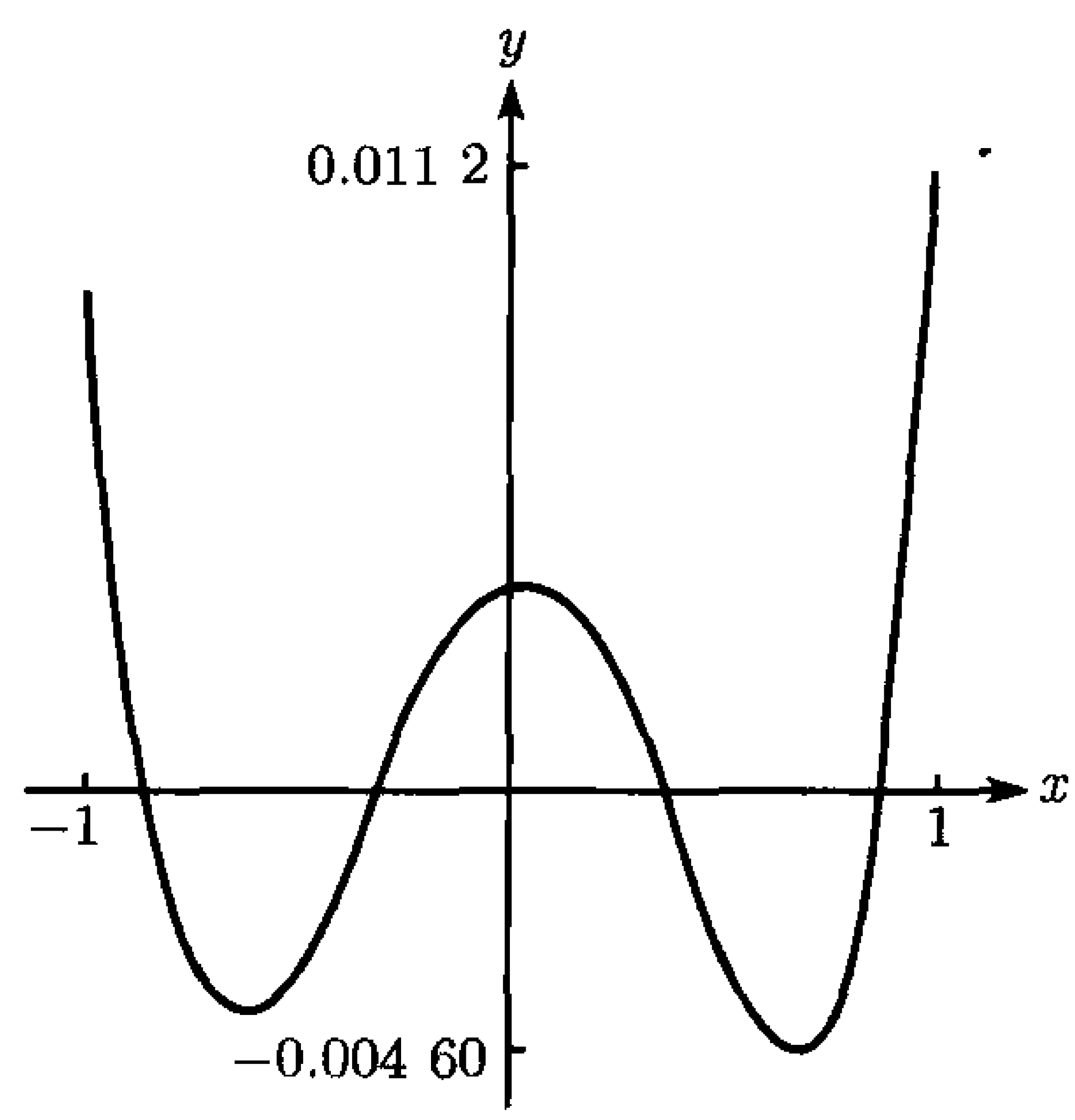


图 4-20 e^x 的三次最小二乘逼近的误差

表 4-11 e^x 的三次逼近的误差

逼近	极大误差	均方根误差
泰勒 $t_3(x)$	0.051 6	0.014 5
最小二乘 $\ell_3(x)$	0.011 2	0.003 34
切比雪夫 $c_3(x)$	0.006 66	0.003 84
极小极大 $m_3(x)$	0.005 53	0.003 88

4.7.3 一般的最小二乘逼近

由于多种原因, 我们通过考虑

$$E(p;f) \equiv \sqrt{\frac{1}{c}\int_a^b w(x)[f(x)-p(x)]^2\mathrm{d}x}, \quad c = \int_a^b w(x)\mathrm{d}x \tag{4.124}$$

来推广 "平均误差" 的概念并试图极小化上式. 它称为**加权极小二乘逼近**(weighted least square approximation), 其中 $w(x)$ 是一个**权函数**(weight function), 即它满足以下假设:

A1. 对于 $a < x < b$, $w(x) > 0$;

A2. 对于所有整数 $n \geqslant 0$

$$\int_a^b w(x)|x|^n\mathrm{d}x < \infty$$

在这些假设下, 倘若再把记号 (4.116) 扩展为

$$(f,g)_w \equiv \int_a^b w(x)f(x)g(x)\mathrm{d}x \tag{4.125}$$

则很容易推广上述利用勒让德多项式的讨论过程. 存在勒让德多项式的推广, 它们满足关键的正交性质 (4.118), 我们在习题 8 和习题 9 中给出一些有关的例子. 人们已经建立了这样的正交多项式的一般理论, 其入门知识, 可参见 Atkinson(1989, 第 4 章).

习题

1. 求出区间 $[0,1]$ 上 $f(x)=\mathrm{e}^x$ 的线性最小二乘逼近. 利用例 4.7.1 的直接方法, 但是选用区间 $[0,1]$.
2. 求出区间 $[0,\frac{1}{2}\pi]$ 上 $f(x)=\sin x$ 的线性最小二乘逼近. 利用例 4.7.1 的直接方法, 但是选用区间 $[0,\frac{1}{2}\pi]$.
3. 对于区间 $[-1,1]$ 上的下列函数, 求出它们的次数 $\leqslant 4$ 的最小二乘逼近, 写成 (4.123) 的形式. 可以利用数值积分 (参见第 5 章) 或者利用像 MAPLE 和 MATHMATICA 那样的符号积分算子求出系数 (f,P_j). 再画出区间 $[-1,1]$ 上误差 $f(x)-l_n(x)$ 的图像.
   (a) $f(x)=\sin(\pi x)$;
   (b) $f(x)=\ln(1+x^2)$;
   (c) $f(x)=\tan^{-1}x$;
   (d) $f(x)=\mathrm{e}^t,\quad t=\frac{1}{2}(x+1)$;
   (e) $f(x)=\cos\left[\frac{1}{4}\pi(x+1)\right]$.
4. 对于 $0\leqslant i,j\leqslant 2$, 证明 (4.118).
5. 对于 $n=1,2,3$, 证明 (4.119).
6. 对于每个 $n\geqslant 0$ 以及适当选取的系数 $\{c_{j,n}\}$, 证明

$$x^n=\sum_{j=0}^{n}c_{j,n}P_j(x) \tag{4.126}$$

   从证明 $n=0$ 和 $n=1$ 的情况开始, 得到所需的系数 $\{c_{j,n}\}$. 对于 $n\geqslant 2$ 的一般情况, 注意到可以在公式 (4.114) 中求解 x^n, 得到一个包含 $P_n(x)$ 和 $\{1,x,x^2,\cdots,x^{n-1}\}$ 的公式. 然后证明可以设计一个得到 (4.126) 的递推算法. 例如, 对 $n=2$, 利用 $P_2(x)$, 1 和 x, 求出 x^2. 利用 $n=0$ 和 $n=1$ 时的公式 (4.126) 证明 $n=2$ 时的公式 (4.126). 对于 $n=3$, 重复上述过程, 首先利用 $P_3(x)$, 1, x 和 x^2, 求出 x^3. 然后利用 $n=0$, $n=1$ 和 $n=2$ 时的公式 (4.126) 证明 $n=3$ 时的公式 (4.126). 并请参见与之密切相关的 4.5 节中习题 9.
7. 利用习题 6 的结果, 证明 $n=2$ 时的基底性质 (4.120). 将 $p(x)$ 写成下列标准形式

$$p(x)=a_0+a_1x+a_2x^2$$

   从 $p(x)$ 的系数 $\{a_j\}$ 求出 (4.120) 中的系数 $\{\beta_j\}$. 对于 $n=3$ 的情形, 也由同样的过程得到 (4.120).

8. 对于 $-1 < x < 1$, 设 $w(x) = 1/\sqrt{1-x^2}$. 证明切比雪夫多项式 $\{T_k(x) : k \geqslant 0\}$ 满足

$$(T_i, T_j)_w = 0, \quad i \neq j$$

试问这时 $(T_i, T_j)_w$ 是什么?

9. 根据 4.5 节中习题 8, 回顾第二类切比雪夫多项式 $S_n(x)$ 的定义. 设在 $-1 \leqslant x \leqslant 1$ 上 $w(x) = \sqrt{1-x^2}$. 证明正交性质

$$(S_i, S_j)_w = 0, \quad i \neq j$$

10. 用第二类切比雪夫多项式 $\{S_k(x)\}_{k\geqslant 0}$ 代替勒让德多项式 $\{P_k(x)\}_{k\geqslant 0}$, 重复习题 6 中的工作.

第 5 章　数值积分和数值微分

定积分

$$I(f)=\int_a^b f(x)\mathrm{d}x \tag{5.1}$$

在微积分学中被定义为黎曼和(Riemann sums) 的极限值. 可以证明

$$I(f)=F(b)-F(a) \tag{5.2}$$

其中 $F(x)$ 是 $f(x)$ 的任一原函数, 这是微积分基本定理(fundamental theorem of calculus). 许多定积分的值可以利用这个公式计算, 大多数微积分教科书使用这种方法来计算定积分的值. 然而, 大部分定积分其实无法通过 (5.2) 式求值, 因为大部分被积函数 $f(x)$ 的原函数不能用初等函数表示. 例如定积分:

$$\int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x,\quad \int_0^\pi x^\pi \sin(\sqrt{x})\mathrm{d}x \tag{5.3}$$

所以我们需要通过其他方法来计算这样的定积分.

5.1 节讨论两种经典且很常用的近似计算 (5.1) 的数值方法: 梯形法和辛普森(Simpson) 法. 5.2 节分析这两种方法的误差并在此基础上对方法进行改进. 5.3 节介绍另一种数值积分方法: 高斯求积法. 高斯求积法的形式比梯形法和辛普森法复杂, 但是在计算量相当的情况下它的精度几乎总是最高. 5.4 节讨论了数值微分. 我们导出一些简易的方法, 并且分析这些方法对舍入误差的灵敏度.

5.1　梯形法和辛普森法

许多计算积分

$$I(f)=\int_a^b f(x)\mathrm{d}x$$

的近似公式的核心思想是用一个易于求积分的近似函数代替 $f(x)$. 本节将研究基于线性插值和二次插值的方法.

$f(x)$ 的线性插值多项式为

$$P_1(x)=\frac{(b-x)f(a)+(x-a)f(b)}{b-a}$$

其中插值节点为 a 和 b(如图 5-1). $P_1(x)$ 在区间 $[a,b]$ 上的积分值等于图 5-1 阴影部分梯形的面积, 其值为

$$T_1(f) = (b-a)\left[\frac{f(a)+f(b)}{2}\right] \tag{5.4}$$

当 $f(x)$ 在区间 $[a,b]$ 上近似于线性函数时, 可以通过这一方法估计积分 $I(f)$ 的值.

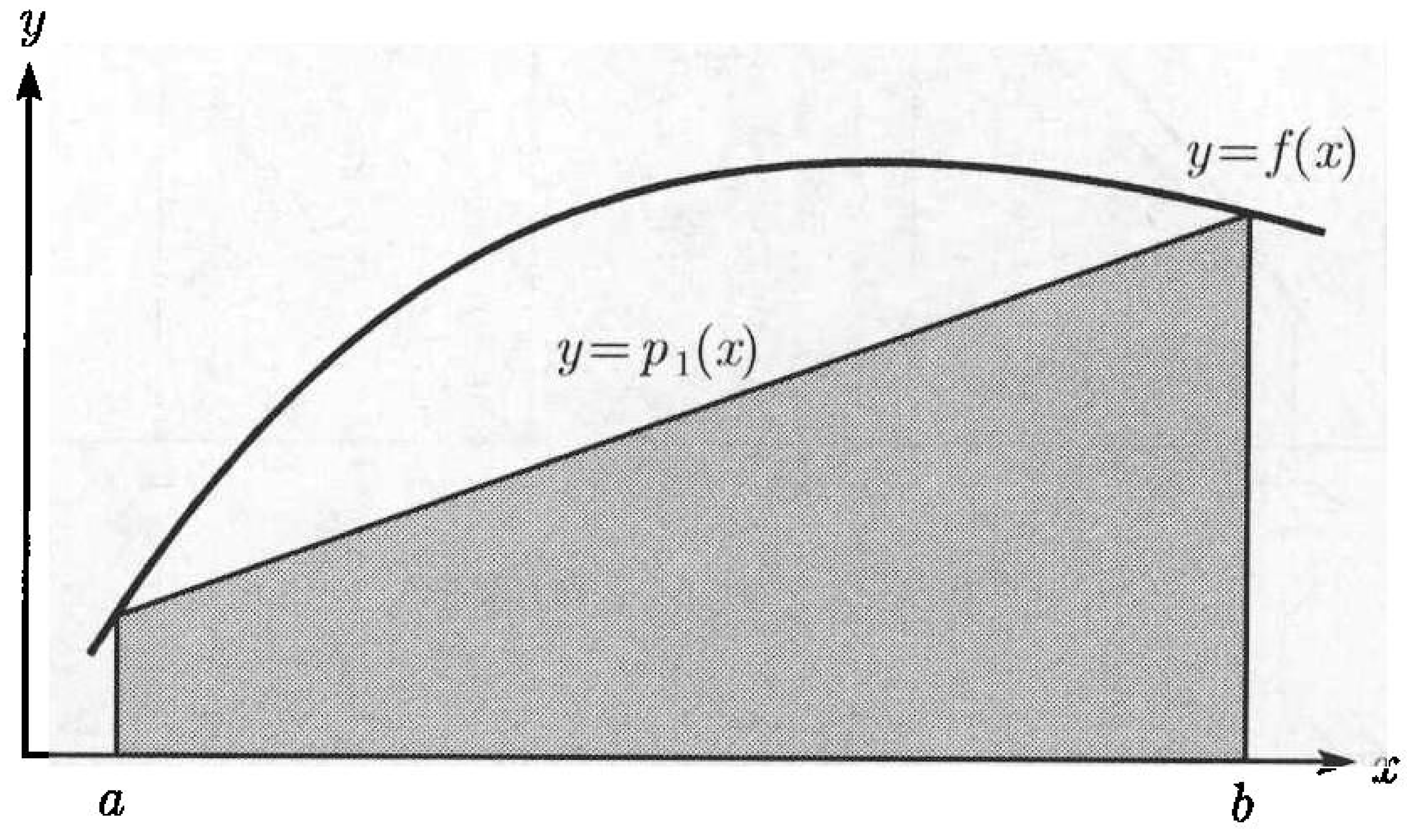


图 5-1 梯形法 (5.4) 的图解

例 5.1.1 近似计算积分

$$I = \int_0^1 \frac{\mathrm{d}x}{1+x} \tag{5.5}$$

的值. 其精确值为 $I = \ln(2) \doteq 0.693\ 147$. 利用公式 (5.4), 得到

$$T_1(f) = \frac{1}{2}\left[1+\frac{1}{2}\right] = \frac{3}{4} = 0.75 \tag{5.6}$$

误差为

$$I - T_1 \doteq -0.056\ 9 \qquad \blacksquare(5.7)$$

当 $f(x)$ 在 $[a,b]$ 上不近似于线性函数时, 为了提高 (5.4) 中近似值 $T_1(f)$ 的精度, 可将区间 $[a,b]$ 分割成若干个子区间, 并在每个子区间上根据 (5.4) 式计算积分. 当这些子区间充分小时, $f(x)$ 在每个子区间上近似于线性的. 这种做法的思想在图 5-2 中说明.

例 5.1.2 在两个等长的子区间上, 利用 $T_1(f)$ 计算例 5.1.1 中的积分值.

$$I = \int_0^{1/2} \frac{\mathrm{d}x}{1+x} + \int_{1/2}^1 \frac{\mathrm{d}x}{1+x}$$

$$I \doteq \frac{1}{2}\left[\frac{1+\frac{2}{3}}{2}\right] + \frac{1}{2}\left[\frac{\frac{2}{3}+\frac{1}{2}}{2}\right]$$

$$T_2 \doteq \frac{17}{24} \doteq 0.708\ 33 \tag{5.8}$$

$$I - T_2 \doteq -0.015\ 2 \tag{5.9}$$

T_2 的误差大约是 (5.7) 中 T_1 的误差的 $\frac{1}{4}$. ■

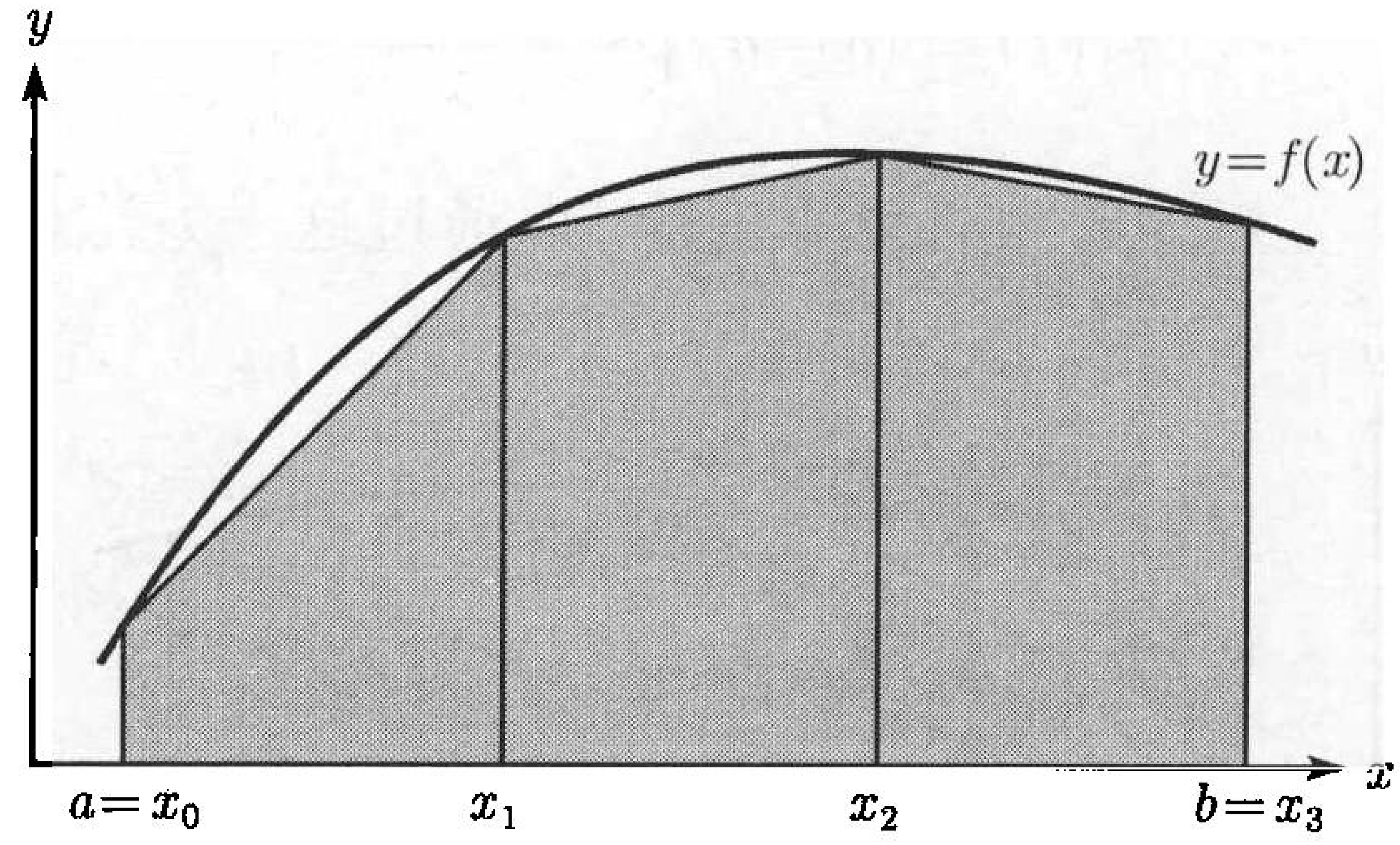


图 5-2　$T_3(f)$ 的图解

当使用多个等长的子区间时, 我们可以导出一个一般公式来简化计算. 设 n 表示子区间的个数, 令

$$h = \frac{b-a}{n}$$

为每个子区间的长度. 子区间的端点为

$$x_j = a + jh, \quad j = 0, 1, \cdots, n$$

将原积分拆成 n 个积分的和

$$\begin{aligned} I(f) &= \int_a^b f(x)\mathrm{d}x = \int_{x_0}^{x_n} f(x)\mathrm{d}x \\ &= \int_{x_0}^{x_1} f(x)\mathrm{d}x + \int_{x_1}^{x_2} f(x)\mathrm{d}x + \cdots + \int_{x_{n-1}}^{x_n} f(x)\mathrm{d}x \end{aligned} \tag{5.10}$$

利用 (5.4) 式近似计算每个积分, 并注意每个子区间 $[x_{j-1}, x_j]$ 的长度为 h, 则有

$$\begin{aligned} I(f) \approx& h\left[\frac{f(x_0)+f(x_1)}{2}\right] + h\left[\frac{f(x_1)+f(x_2)}{2}\right] \\ &+ h\left[\frac{f(x_2)+f(x_3)}{2}\right] + \cdots + h\left[\frac{f(x_{n-1})+f(x_n)}{2}\right] \end{aligned}$$

将右端项合并, 可得到较简洁形式

$$T_n(f) = h\left[\frac{1}{2}f(x_0) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right] \tag{5.11}$$

这种方法称为**梯形数值积分法**(trapezoidal numerical integration rule). 下标 n 表示所用子区间的个数, $x_0, x_1, \cdots, x_n$ 称为数值积分的**节点**(node points).

在给出 $T_n(f)$ 的若干数值例子之前, 我们先讨论如何选取 n. 随着 n 的值不断增加, $T_n(f)$ 会越来越精确地逼近 $I(f)$. 那么应该如何选取序列 n 的值呢? 如果 n 不断地翻倍, 则每个 $T_{2n}(f)$ 所需的函数值就将包括之前 $T_n(f)$ 中所有已用过的函数值. 因此, 把 n 翻倍的方法将保证在新的计算中可利用前面已算出的函数值, 因而相比其他方法, 梯形积分法的计算量比较小. 下面通过 $T_2(f)$ 和 $T_4(f)$ 来说明当 n 翻倍时函数值是如何被重复使用的.

$$T_2(f) = h\left[\frac{f(x_0)}{2} + f(x_1) + \frac{f(x_2)}{2}\right] \tag{5.12}$$

其中

$$h = \frac{b-a}{2}, \quad x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b$$

还有

$$T_4(f) = h\left[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + f(x_3) + \frac{f(x_4)}{2}\right] \tag{5.13}$$

其中

$$h = \frac{b-a}{4}, \quad x_0 = a, \quad x_1 = \frac{3a+b}{4}, \quad x_2 = \frac{a+b}{2}, \quad x_3 = \frac{a+3b}{4}, \quad x_4 = b$$

在 (5.13) 中, 只需要计算 $f(x_1)$ 和 $f(x_3)$ 的值, 其他的函数值在计算 (5.12) 时已经得到. 由于这个原因及其他一些原因, 我们所有关于 $T_n(f)$ 的例子都是基于将 n 翻倍的.

例 5.1.3 对以下三个积分给出 $T_n(f)$ 的计算:

$$I^{(1)} = \int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x \doteq 0.746\,824\,132\,812\,427 \tag{5.14}$$

$$I^{(2)} = \int_0^4 \frac{\mathrm{d}x}{1+x^2} = \tan^{-1}(4) \doteq 1.325\,817\,663\,668\,03 \tag{5.15}$$

$$I^{(3)} = \int_0^{2\pi} \frac{\mathrm{d}x}{2+\cos(x)} = \frac{2\pi}{\sqrt{3}} \doteq= 3.627\,598\,728\,468\,44 \tag{5.16}$$

结果如表 5-1 所示. 这里只给出了误差 $I(f) - T_n(f)$, 因为这个数值是考察 $T_n(f)$ 收敛于 $I(f)$ 速度快慢的主要依据. “比率” 这一列给出了逐次误差的比率, 即 n 翻倍后, 两误差递减的比值.

如表 5-1 所示, 当 n 翻倍时积分 $I^{(1)}$ 和 $I^{(2)}$ 的误差递减的比值大约为 4. 在第三个积分值 $I^{(3)}$ 的计算过程中, 收敛非常迅速. $n = 32, 64$ 和 128 时的误差已接近于计算机的舍入误差界 (大约 16 位十进制数字), 所以在表中用符号 * 表示. 这一现象产生的原因将在下一节中进行讨论. ■

表 5-1　梯形法的例子

n	$I^{(1)}$		$I^{(2)}$		$I^{(3)}$	
	误差	比率	误差	比率	误差	比率
2	1.55E−2		−1.33E−1		−5.61E−1	
4	3.84E−3	4.02	−3.59E−3	37.0	−3.76E−2	14.9
8	9.59E−4	4.01	5.64E−4	−6.37	−1.93E−4	195.0
16	2.40E−4	4.00	1.44E−4	3.92	−5.19E−9	37 600.0
32	5.99E−5	4.00	3.60E−5	4.00	*	
64	1.50E−5	4.00	9.01E−6	4.00	*	
128	3.74E−6	4.00	2.25E−6	4.00	*	

MATLAB 程序　我们给出基于梯形法的程序. 根据用户提供的 n_0, 对 $n = n_0$, $2n_0, 4n_0, \cdots, 256n_0$, 分别计算 $T_n(f)$. 当 n 翻倍为 $2n$ 时, $T_n(f)$ 中出现的所有函数值都被用来计算 $T_{2n}(f)$. 我们还允许选择不同的被积函数, 在子程序 f 中用户可定义被积函数. 程序的注释说明程序的结构.

```
function [integral,difference,ratio]=trapezoidal(a,b,n0,index_f)
%
% This uses the trapezoidal rule with n subdivisions to
% integrate the function f over the interval [a,b].  The
% values of n used are
%   n = n0,2*n0,4*n0,...,256*n0
% The value of n0 must be a positive integer.
% The corresponding numerical integrals are returned in the
% vector integral.  The differences of successive numerical
% integrals are returned in the vector difference:
%   difference(i) = integral(i)-integral(i-1), i=2,...,9
% The entries in ratio give the rate of decrease in these
% differences.
%
% In using this program, define the integrand using the
% function given below.  The parameter index_f allows the
% user to do calculations with multiple integrands.

% Initialize output vectors.
integral = zeros(9,1);
difference = zeros(9,1);
ratio = zeros(9,1);

% Initialize for trapezoidal rule.
sumend = (f(a,index_f) +f(b,index_f))/2;
sum = 0;

% Initialize for case of n0 > 2.
```

```
if(n0 > 2)
  h = (b-a)/n0;
  for i=2:2:n0-2
    sum = sum + f(a+i*h,index_f);
  end
end

% Calculate the numerical integrals, doing each
% by appropriately modifying the preceding case.
for i=1:9
  n = n0*2^(i-1);
  h = (b-a)/n;
  for k=1:2:n-1
    sum = sum + f(a+k*h,index_f);
  end
  integral(i) = h*(sumend + sum);
end

% Calculate the differences of the successive
% trapezoidal rule integrals and the ratio
% of decrease in these differences.
difference(2:9) = integral(2:9)-integral(1:8);
ratio(3:9) = difference(2:8)./difference(3:9);

function f_value = f(x,index)
%
% This defines the integrand.

switch index
case 1
 f_value = exp(-x.^2);
case 2
 f_value = 1 ./(1+x.^2);
case 3
 f_value = 1 ./(2+cos(x));
end
```

辛普森法

为了改进 (5.4) 给出的 $T_1(f)$ 的精度, 可用二次插值多项式来逼近区间 $[a,b]$ 上的 $f(x)$. 设 $P_2(x)$ 为节点 a, $c=(a+b)/2$ 和 b 上 $f(x)$ 的二次插值多项式. 利用这个多项式来逼近 $I(f)$, 得到

$$
\begin{aligned}
I(f) \approx \int_a^b p_2(x)\mathrm{d}x = \int_a^b \bigg[&\frac{(x-c)(x-b)}{(a-c)(a-b)}f(a) \\
&+ \frac{(x-a)(x-b)}{(c-a)(c-b)}f(c) + \frac{(x-a)(x-c)}{(b-a)(b-c)}f(b)\bigg]\mathrm{d}x
\end{aligned} \tag{5.17}
$$

这个积分值可以直接求出, 更简单的算法是先引入 $h=(b-a)/2$, 然后将积分作变量替换. 下面通过计算其中的第一项来说明具体的计算过程. 令 $u=x-a$, 则

$$\begin{aligned}\int_a^b \frac{(x-c)(x-b)}{(a-c)(a-b)}\mathrm{d}x &= \frac{1}{2h^2}\int_a^{a+2h}(x-c)(x-b)\mathrm{d}x \\ &= \frac{1}{2h^2}\int_0^{2h}(u-h)(u-2h)\mathrm{d}u \\ &= \frac{1}{2h^2}\left[\frac{u^3}{3}-\frac{3}{2}u^2h+2h^2u\right]_0^{2h} = \frac{h}{3}\end{aligned}$$

由 (5.17) 式可得

$$S_2(f)=\frac{h}{3}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right] \tag{5.18}$$

图 5-3 给出此方法的图解.

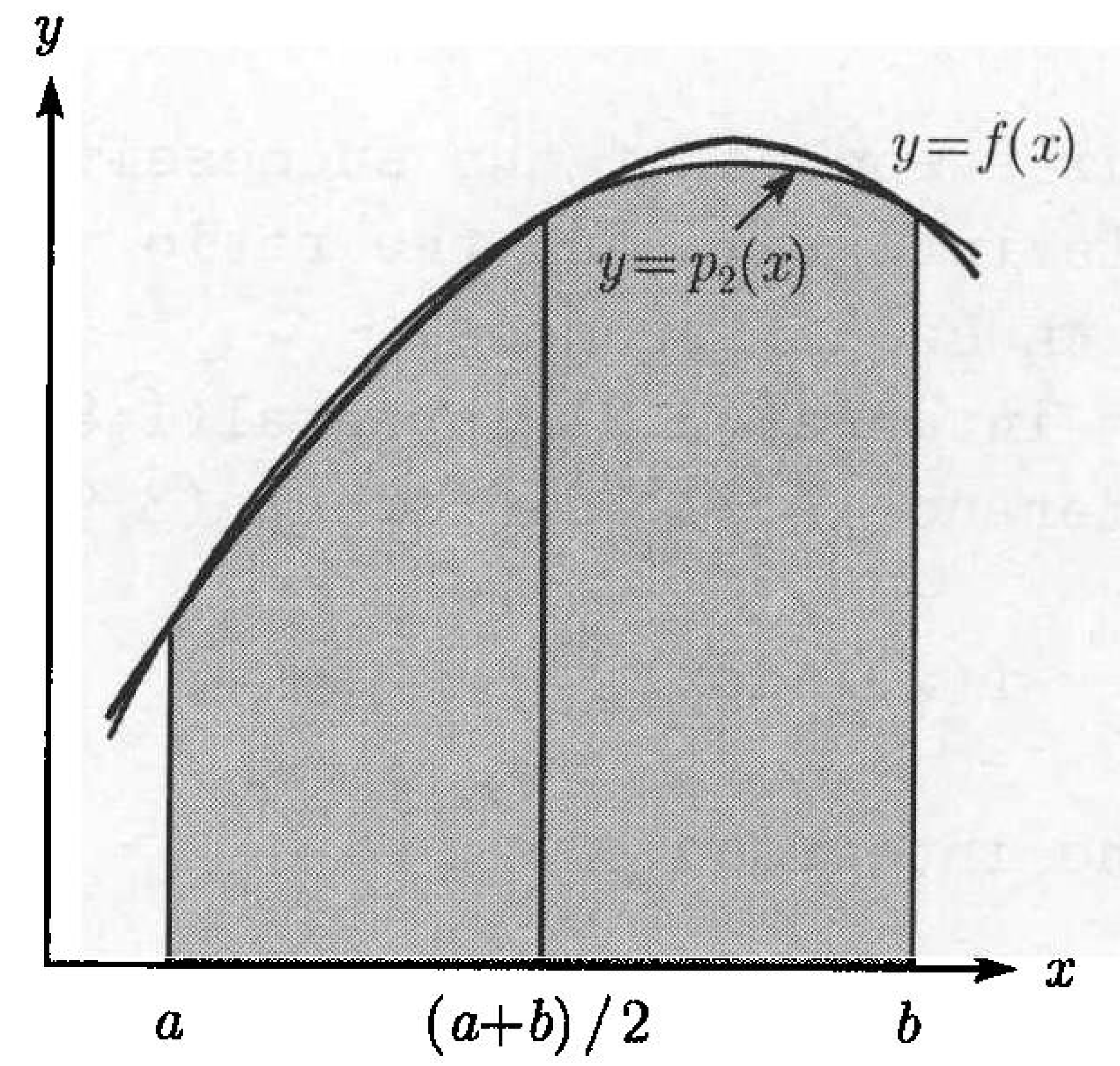


图 5-3　辛普森法图解

例 5.1.4　考虑前面 (5.5) 中的积分

$$I=\int_0^1 \frac{\mathrm{d}x}{1+x}$$

则 $h=(b-a)/2=\frac{1}{2}$, 并且

$$S_2=\frac{1/2}{3}\left[1+4\left(\frac{2}{3}\right)+\frac{1}{2}\right]=\frac{25}{36}\doteq 0.694\ 44 \tag{5.19}$$

其误差为

$$I-S_2=\ln(2)-S_2\doteq -0.001\ 30 \tag{5.20}$$

我们将 S_2 的精度与梯形法中 T_2 的精度相比较, 因为 S_2 和 T_2 需要计算的函数值的个数相同. S_2 中的误差大约是 (5.9) 式中 T_2 的误差的 1/12. ■

如果 $f(x)$ 在 $[a,b]$ 上近似于二次函数, 方法 $S_2(f)$ 可以精确地逼近 $I(f)$ 的值. 否则, 我们可以参照梯形法进行改进. 设 n 为偶数, $h=(b-a)/n$, 定义 $f(x)$ 求值的节点如下:

$$x_j = a + jh, \quad j = 0, 1, \cdots, n \tag{5.21}$$

与 (5.10) 的思想大致一样, 但这里将区间 $[a,b]=[x_0,x_n]$ 分成较大的子区间, 使每个子区间包括三个插值节点. 因此

$$\begin{aligned} I(f) &= \int_a^b f(x)\mathrm{d}x = \int_{x_0}^{x_n} f(x)\mathrm{d}x \\ &= \int_{x_0}^{x_2} f(x)\mathrm{d}x + \int_{x_2}^{x_4} f(x)\mathrm{d}x + \cdots + \int_{x_{n-2}}^{x_n} f(x)\mathrm{d}x \end{aligned}$$

用 (5.18) 分别近似计算每个积分, 得到

$$\begin{aligned} I(f) \approx& \frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)] + \frac{h}{3}[f(x_2)+4f(x_3)+f(x_4)] + \cdots \\ &+ \frac{h}{3}[f(x_{n-2})+4f(x_{n-1})+f(x_n)] \end{aligned}$$

简化后得,

$$\begin{aligned} S_n(f) =& \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_4)+\cdots \\ &+ 2f(x_{n-2})+4f(x_{n-1})+f(x_n)] \end{aligned} \tag{5.22}$$

这种方法称为**辛普森法**(Simpson's rule), 两个多世纪以来, 这种方法是数值积分中最常用的一种方法. 指标 n 表示区间的剖分数, 并用以确定积分的节点 $x_0, \cdots, x_n$.

例 5.1.5 前面我们使用了梯形法计算 (5.14) 到 (5.16) 的积分值. 这里, 我们利用辛普森法对这些积分重新计算, 所得的结果如表 5-2 所示. 积分 $I^{(1)}$ 和 $I^{(2)}$ 的误差递减的比值大约为 16. 积分 $I^{(3)}$ 的误差趋于零的速度则更快. 下一节将解释产生这一现象的原因. ■

表 5-2 辛普森法的例子

n	$I^{(1)}$		$I^{(2)}$		$I^{(3)}$	
	误差	比率	误差	比率	误差	比率
2	−3.56E−4		8.66E−2		−1.26	
4	−3.12E−5	11.4	3.95E−2	2.2	1.37E−1	−9.2
8	−1.99E−6	15.7	1.95E−3	20.3	1.23E−2	11.2
16	−1.25E−7	15.9	4.02E−6	485.0	6.43E−5	191.0
32	−7.79E−9	16.0	2.33E−8	172.0	1.71E−9	37 600.0
64	−4.87E−10	16.0	1.46E−9	16.0	*	
128	−3.04E−11	16.0	9.15E−11	16.0	*	

MATLAB 程序　这里给出辛普森法的程序. 根据用户提供的 n_0, 对 $n = n_0, 2n_0, 4n_0, \cdots, 256n_0$, 计算 $S_n(f)$. 当把 n 翻倍为 $2n$ 时, $S_n(f)$ 中出现的所有函数值都被用来计算 $S_{2n}(f)$. 我们也允许选择不同的被积函数, 用户可在子程序 f 中定义被积函数. 程序的注释说明程序的结构.

```
function [integral,difference,ratio]=simpson(a,b,n0,index_f)
%
% This uses Simpson's rule with n subdivisions to integrate the
% function f over the interval [a,b].  The values of n used are
%    n = n0,2*n0,4*n0,...,256*n0
% The value of n0 MUST be a positive even integer.
% The corresponding numerical integrals are returned in the
% vector integral.  The differences of successive numerical
% integrals are returned in the vector difference:
%    difference(i) = integral(i)-integral(i-1), i=2,...,9
% The entries in ratio give the rate of decrease in these
% differences.
%
% In using this program, define the integrand using the
% function given below.  The parameter index_f allows the
% user to do calculations with multiple integrands.

% Initialize output vectors.
integral = zeros(9,1);
difference = zeros(9,1);
ratio = zeros(9,1);

% Initialize for Simpson integration.
sumend = f(a,index_f) + f(b,index_f);
sumodd = 0;
sumeven = 0;

% Initialize for case of n0 > 2.
if(n0 > 2)
  h = (b-a)/n0;
  for i=2:2:n0-2
    sumeven = sumeven + f(a+i*h,index_f);
  end
end

% Calculate the numerical integrals, doing each
% by appropriately modifying the preceding case.
for i=1:9
n = (n0)*(2^(i-1));
h = (b-a)/n;
```

```
sumeven = sumeven + sumodd;
sumodd = 0;
for k=1:2:n-1
  sumodd = sumodd + f(a+k*h,index_f);
end

  integral(i) = h*(sumend + 4*sumodd + 2*sumeven)/3;
end

% Calculate the differences of the successive
% Simpson rule integrals and the ratio
% of decrease in these differences.
difference(2:9) = integral(2:9)-integral(1:8);
ratio(3:9) = difference(2:8)./difference(3:9);

function f_value = f(x,index)
%
% This defines the integrand.

switch index
case 1
 f_value = exp(-x.^2);
case 2
 f_value = 1 ./(1+x.^2);
case 3
 f_value = 1 ./(2+cos(x));
end
```

习题

1. 对 (5.5) 中的积分 I 计算 $T_4(f)$ 和 $S_4(f)$. 计算误差 $I - T_4(f)$ 和 $I - S_4(f)$, 并分别将它们与误差 $I - T_2(f)$ 和 $I - S_2(f)$ 进行比较.
2. 根据教材中给出的梯形法程序, 对 $n = 2, 4, 8, \cdots, 512$, 列表记录以下积分的近似值 $T_n(f)$, 并且求出误差和误差减少的比率.

   (a) $\int_0^{\pi} \mathrm{e}^x \cos(4x)\mathrm{d}x = \frac{\mathrm{e}^{\pi}-1}{17}$

   (b) $\int_0^1 x^{5/2}\mathrm{d}x = \frac{2}{7}$

   (c) $\int_0^5 \frac{\mathrm{d}x}{1+(x-\pi)^2} = \tan^{-1}(5-\pi) + \tan^{-1}(\pi)$

   (d) $\int_{-\pi}^{\pi} \mathrm{e}^{\cos(x)}\mathrm{d}x \doteq 7.954\,926\,521\,012\,84$

   (e) $\int_0^{\pi/4} \mathrm{e}^{\cos(x)}\mathrm{d}x \doteq 1.939\,734\,850\,623\,65$

   (f) $\int_0^1 \sqrt{x}\mathrm{d}x = \frac{2}{3}$

3. 用辛普森法重做习题 2.

4. 用梯形法和辛普森法, 求出下列函数 $f(x)$ 的图像 $y=f(x)$ 在所给的区间上与 x 轴所围成的区域的面积近似值, 取 $n=4,8,\cdots,512$.

(a) $f(x)=\mathrm{e}^{-x^2},\quad 0\leqslant x\leqslant 10$

(b) $f(x)=\tan^{-1}(1+x^2),\quad 0\leqslant x\leqslant 2$

(c) $f(x)=\sqrt{x}\mathrm{e}^x,\quad 0\leqslant x\leqslant 1$

5. 我们已经知道, 由函数 $y=f(x)$ 所表示的曲线在区间 $[a,b]$ 上的长度可用如下的积分表示:

$$I(f)=\int_a^b\sqrt{1+[f'(x)]^2}\mathrm{d}x$$

用梯形法和辛普森法计算下列曲线的长度, 取 $n=4,8,\cdots,512$.

(a) $f(x)=\sin(\pi x),\quad 0\leqslant x\leqslant 1$

(b) $f(x)=\mathrm{e}^x,\quad 0\leqslant x\leqslant 1$

(c) $f(x)=\mathrm{e}^{x^2},\quad 0\leqslant x\leqslant 1$

6. 在不同的区间 $[0,b]$ 上, 用数值方法计算积分

$$I=\int_0^b\frac{\mathrm{d}x}{2+\sin x}$$

进行实验. 特别地, 取 $b=\frac{1}{2}\pi,\pi,2\pi,10$. 分别利用梯形法 T_n 和辛普森法 S_n, 并选取一些不同的 n 值. 当 $-\frac{1}{2}\pi<b<\frac{1}{2}\pi$ 时, 精确的积分值为

$$\frac{2\sqrt{3}}{3}\tan^{-1}\left(\frac{\sqrt{3}}{3}\left(2\tan\frac{b}{2}+1\right)\right)-\frac{\sqrt{3}}{9}\pi$$

讨论数值结果, 包括所选取的数值方法的收敛速度.

7. 计算 (5.17) 中的后两个积分, 并验证 (5.18) 式.

8. **(a)** 作为 $I(f)=\int_a^b f(x)\mathrm{d}x$ 的一种不同的近似方法, 在整个区间 $a\leqslant x\leqslant b$ 上用常数 $f[(a+b)/2]$ 来代替 $f(x)$. 证明由此导出的数值积分公式为

$$M_1(f)=(b-a)f\left(\frac{a+b}{2}\right)$$

画图解释这个近似方法.

(b) 仿照推导梯形法 (5.11) 和辛普森法 (5.22) 的做法, 把 (a) 的数值积分公式推广为

$$M_n(f)=h[f(x_1)+f(x_2)+\cdots+f(x_n)]$$

其中 $h=(b-a)/n$, 且

$$x_j=a+\left(j-\frac{1}{2}\right)h,\quad j=1,\cdots,n$$

这种方法称为*中点法*(midpoint rule), 常可用它来代替梯形法.

(c) 用公式 $M_1(f)$ 和 $M_2(f)$ 来计算 (5.5) 中的积分 $I=\int_0^1\mathrm{d}x/(1+x)$. 将其误差分别与 (5.7) 中的 T_1 和 (5.9) 中的 T_2 进行比较.

9. 用中点法重做习题 2.

10. **(a)** 作为 $I(f)=\int_a^b f(x)\mathrm{d}x$ 的另一种近似方法, 用 $f(x)$ 在下列 5 个点

$$x_j=a+jh,\quad j=0,1,2,3,4,\quad h=\frac{b-a}{4}$$

处的四次插值多项式来近似代替 $f(x)$. 证明由此导出的近似公式是

$$B_4(f)=\frac{2h}{45}[7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)]$$

这个方法被称为*布尔法*(Boole's rule), 与辛普森法和梯形法一样, 这个方法同样也可以在子区间上使用.

(b) 用公式 $B_4(f)$ 计算 $I=\int_0^1 \mathrm{d}x/(1+x)$ 的值, 并且将计算结果与习题 1 中的结果进行比较.

11. 数值积分公式的*精度*(degree of precision) 定义如下: 如果求积公式对于所有次数 $\leqslant r$ 的多项式误差为零, 且对于某个次数为 $r+1$ 的多项式误差非零, 则我们称这一求积公式的精度为 r. 证明下列求积公式的精度为所给出的 r.

(a) $M_1(f):r=1$ ($M_1(f)$ 的定义见习题 8)

(b) $T_1(f):r=1$

(c) $S_2(f):r=3$

(d) $B_4(f):r=5$ ($B_4(f)$ 的定义见习题 10)

当各种下标变为 n 时, 精度是否不变?

提示: 解题中, 对 $I(f)=\int_0^b f(x)\mathrm{d}x$, 只考虑适当的 b, 以及 $f(x)=1,x,x^2$, 等等.

12. 确定近似公式

$$\int_0^1 f(x)\mathrm{d}x\approx\frac{1}{4}f(0)+\frac{3}{4}f\left(\frac{2}{3}\right)$$

的精度.

13. 令

$$I_h(f)=\frac{3h}{4}[f(0)+3f(2h)]$$

$I_h(f)\approx\int_0^{3h} f(x)\mathrm{d}x$ 的精度为多少?

提示: 考虑 $f(x)=1,x,x^2,x^3$, 等等.

14. 用 $f(x)$ 在 $x=-\frac{1}{3}$ 和 $x=\frac{1}{3}$ 处的线性插值多项式 $P_1(x)$ 代替 $f(x)$ 来近似计算 $I(f)=\int_{-1}^1 f(x)\mathrm{d}x$, 给出数值积分公式. 它的精度为多少?

15. 考虑近似求积公式

$$I(f)=\int_{-1}^1 f(x)\mathrm{d}x\approx f(-\beta)+f(\beta)$$

其中 $0<\beta\leqslant 1$. 证明对任意的 β, 这一求积公式的精度至少为 1. 适当选取 β, 使所得的求积公式精度大于 1. 该求积公式的精度为多少?

16. 用 $f(x)$ 在 $x=0$ 和 $x=h$ 处的线性插值多项式 $P_1(x)$ 代替 $f(x)$ 来近似计算 $I(f)=\int_0^{2h} f(x)\mathrm{d}x$, 给出最终的数值积分公式. 它的精度为多少?

5.2 误 差 公 式

在前一节中, 除一种情况之外的所有被积函数的数值结果显示, 梯形法和辛普森法的误差有规律地变化. 我们通过求积方法的误差公式来解释这些规律. 这些误差公式可以使我们更好地理解求积方法的优点和缺点, 并且可以在此基础上改进求积方法. 我们先研究梯形法的误差.

定理 5.2.1 设 $f(x)$ 在 $[a,b]$ 上有连续的二阶导数, 且 n 为正整数. 则用 (5.11) 中的梯形法 $T_n(f)$ 来计算积分

$$I(f)=\int_a^b f(x)\mathrm{d}x$$

的误差为

$$E_n^T(f)\equiv I(f)-T_n(f)=\frac{-h^2(b-a)}{12}f''(c_n) \tag{5.23}$$

其中 c_n 是 $[a,b]$ 上的未知点, $h=(b-a)/n$.

这里略去定理的证明, 其中的一部分在后文中给出. 一般而言, 通过界定 $|f''(c_n)|$ 在区间 $[a,b]$ 上可能的最大值, 就可以根据 (5.23) 式确定 $T_n(f)$ 的误差界. 我们通过下面的例子来说明这一点. 注意到公式 $E_n^T(f)$ 与表 5-1 中积分 $I^{(1)}$ 和 $I^{(2)}$ 的误差性态是一致的. 当 n 翻倍时, h 减半, h^2 项递减的比值是 4. 这与表 5-1 中梯形法误差减少的比值完全一致.

例 5.2.2 在前一节关于 (5.5) 的例子中,

$$I=\int_0^1\frac{\mathrm{d}x}{1+x}=\ln(2)$$

这里 $f(x)=1/(1+x)$, $[a,b]=[0,1]$, 且 $f''(x)=2/(1+x)^3$. 由 (5.23) 得

$$E_n^T(f)=\frac{-h^2}{12}f''(c_n),\quad 0\leqslant c_n\leqslant 1,\quad h=\frac{1}{n} \tag{5.24}$$

因为 c_n 是未知的, 所以我们不能精确地计算这个公式的值. 但是通过考察 $|f''(c_n)|$ 可能的最大值, 可以求出其误差界. 在 $[a,b]=[0,1]$ 上界定 $|f''(x)|$:

$$\max_{0\leqslant x\leqslant 1}\frac{2}{(1+x)^3}=2$$

则

$$\left|E_n^T(f)\right|\leqslant\frac{h^2}{12}(2)=\frac{h^2}{6} \tag{5.25}$$

对于 $n=1$ 和 $n=2$, 我们有

$$\left|E_1^T(f)\right|\leqslant\frac{1}{6}\doteq 0.167,\quad \left|E_2^T(f)\right|\leqslant\frac{(1/2)^2}{6}\doteq 0.041\,7$$

将这两个结果与 (5.7) 和 (5.9) 中的实际的误差进行比较, 可以发现这些误差界是实际误差的 2 到 3 倍. ■

根据定理 5.2.1 的假设条件可以推断梯形法可能存在的缺点. 如果 $f(x)$ 在 $[a,b]$ 上不存在连续的二阶导数, 试问 $T_n(f)$ 的收敛速度是否会减慢? 对于一些函数, 尤其是一阶导数不连续的函数, 其答案是肯定的. 这个问题留在习题 14 中探讨.

5.2.1 梯形法误差的渐近估计

因为 $f''(c_n)$ 是未知的, 所以误差公式 (5.23) 只能用于界定误差. 我们通过更仔细地研究误差公式可改进上述公式.

(5.23) 的证明要点在于证明当 $n=1$ 时, 在区间 $[a,a+h]$ 上有

$$\int_{\alpha}^{\alpha+h} f(x)\mathrm{d}x - h\left[\frac{f(\alpha)+f(\alpha+h)}{2}\right] = -\frac{h^3}{12}f''(c) \tag{5.26}$$

其中 c 为 $[\alpha,\alpha+h]$ 上的某个数. 根据线性插值的误差公式 (4.53) 可以给出公式 (5.26) 的一个简单的证明, 习题 11 中给出了另一种利用泰勒多项式逼近的证明方法. 这里, 我们只给出如何利用 (5.26) 得到定理 5.2.1 中的一般公式 (5.23).

回顾 5.1 节中 (5.10) 及以后所给出的梯形法 $T_n(f)$ 的推导, 得到

$$\begin{aligned} E_n^T(f) &= \int_a^b f(x)\mathrm{d}x - T_n(f) = \int_{x_0}^{x_n} f(x)\mathrm{d}x - T_n(f) \\ &= \int_{x_0}^{x_1} f(x)\mathrm{d}x - h\left[\frac{f(x_0)+f(x_1)}{2}\right] + \int_{x_1}^{x_2} f(x)\mathrm{d}x - h\left[\frac{f(x_1)+f(x_2)}{2}\right] \\ &\quad + \cdots + \int_{x_{n-1}}^{x_n} f(x)\mathrm{d}x - h\left[\frac{f(x_{n-1})+f(x_n)}{2}\right] \end{aligned} \tag{5.27}$$

把 (5.26) 应用到 (5.27) 中等式右端的每一项, 则

$$E_n^T(f) = -\frac{h^3}{12}f''(\gamma_1) - \frac{h^3}{12}f''(\gamma_2) - \cdots - \frac{h^3}{12}f''(\gamma_n) \tag{5.28}$$

未知常数 $\gamma_1,\cdots,\gamma_n$ 分别位于子区间

$$[x_0,x_1],[x_1,x_2],\cdots,[x_{n-1},x_n] \tag{5.29}$$

内. 提取 (5.28) 中的公因子, 得到

$$E_n^T(f) = -\frac{h^2}{12}[hf''(\gamma_1)+\cdots+hf''(\gamma_n)] \tag{5.30}$$

可以证明括号中的总和等于 $(b-a)f''(c_n)$, 其中 c_n 为 $[a,b]$ 上的某个数, 从而得到了一般形式 (5.23)(参见习题 12).

为了**估计**(estimate) 梯形法的误差, 我们注意 (5.30) 式括号中的项是积分

$$\int_a^b f''(x)\mathrm{d}x = f'(b) - f'(a) \tag{5.31}$$

的黎曼和. 这个黎曼和是基于对 $[a,b]$ 的划分 (5.29), 当 $n \to \infty$ 时, 黎曼和趋近 (5.31) 中的积分. 用 (5.31) 式来计算 (5.30) 式的右端, 则有

$$E_n^T(f) \approx -\frac{h^2}{12}[f'(b) - f'(a)] \tag{5.32}$$

用 $\tilde{E}_n^T(f)$ 表示这个误差估计. 因为随着 n 的增加, 误差估计的精度不断改进, 所以它称为误差的**渐近估计**(asymptotic estimate). 只要 $f'(x)$ 是可计算的, 我们就可以很容易地计算 $\tilde{E}_n^T(f)$.

例 5.2.3 再次考虑积分 $I = \int_0^1 \frac{\mathrm{d}x}{1+x}$. 我们有 $f'(x) = -1/(1+x)^2$, 并且从 (5.32) 式得到估计

$$\tilde{E}_n^T(f) = -\frac{h^2}{12}\left[\frac{-1}{(1+1)^2} - \frac{-1}{(1+0)^2}\right] = -\frac{h^2}{16}, \quad h = \frac{1}{n} \tag{5.33}$$

对于 $n=1$ 和 $n=2$ 有

$$\tilde{E}_1^T(f) = -\frac{1}{16} = -0.062\ 5, \quad \tilde{E}_2^T(f) \doteq -0.015\ 6$$

这两个估计与 (5.7) 和 (5.9) 中的实际误差十分接近. ■

与前面的误差公式 (5.23) 相比, 估计 $\tilde{E}_n^T(f)$ 有几个实用的优点. 首先, 倘若 $f'(b) - f'(a) \neq 0$, 它确认当 n 翻倍 (即 h 减半) 时, 误差减少的比值是 4. 这与表 5-1 中 $I^{(1)}$ 和 $I^{(2)}$ 的数值结果相吻合. 其次, (5.32) 式显示当 $f'(b) - f'(a) = 0$ 时, $T_n(f)$ 的收敛速度将更快. 这也部分地解释了为什么表 5-1 中 $I^{(3)}$ 的收敛速度非常快, 更进一步的讨论在本节末给出. 最后, 如果将 $\tilde{E}_n^T(f)$ 考虑在内, 则根据 (5.32) 式可以得到更准确的数值积分公式:

$$I(f) - T_n(f) \approx -\frac{h^2}{12}[f'(b) - f'(a)]$$

$$I(f) \approx T_n(f) - \frac{h^2}{12}[f'(b) - f'(a)] \tag{5.34}$$

这种方法称为**修正梯形法**(corrected trapezoidal rule), 记为 $CT_n(f)$.

例 5.2.4 重新计算表 5-1 中的积分 $I^{(1)}$,

$$I = \int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x \doteq 0.746\ 824\ 132\ 812\ 43$$

表 5-3 给出了利用 $T_n(f)$ 和 $CT_n(f)$ 的数值结果, 包括它们的误差和估计

$$\tilde{E}_n^T(f) = \frac{h^2 \mathrm{e}^{-1}}{6}, \quad h = \frac{1}{n}$$

注意 $\tilde{E}_n^T(f)$ 是真实误差的一个十分精确的估计. 同样, 与 $T_n(f)$ 的误差相比, $CT_n(f)$ 的误差收敛于零的速度更快. 当 n 翻倍时, $CT_n(f)$ 的误差减少的比率大约是 16. ■

表 5-3 $CT_n(f)$ 和 $\tilde{E}_n^T(f)$ 的数值例子

n	$I - T_n(f)$	$\tilde{E}_n(f)$	$CT_n(f)$	$I - CT_n(f)$	比率
2	1.545E−2	1.533E−2	0.746 698 561 877	1.26E−4	
4	3.840E−3	3.832E−3	0.746 816 175 313	7.96E−6	15.8
8	9.585E−4	9.580E−4	0.746 823 634 224	4.99E−7	16.0
16	2.395E−4	2.395E−4	0.746 824 101 633	3.12E−8	16.0
32	5.988E−5	5.988E−5	0.746 824 130 863	1.95E−9	16.0
64	1.497E−5	1.497E−5	0.746 824 132 690	2.22E−10	16.0

5.2.2 辛普森法的误差公式

前面关于梯形法误差的讨论也适用于辛普森法. 我们不加证明地给出下面的定理.

定理 5.2.5 假设 $f(x)$ 在 $[a,b]$ 上有四阶连续的导数, 且 n 为正偶数. 则辛普森法的误差为

$$E_n^S(f) = I(f) - S_n(f) = -\frac{h^4(b-a)}{180} f^{(4)}(c_n) \tag{5.35}$$

其中 c_n 为 $[a,b]$ 上的未知点, $h = (b-a)/n$. 误差可进一步用下列渐近误差公式来估计:

$$\tilde{E}_n^S(f) = -\frac{h^4}{180} [f'''(b) - f'''(a)] \tag{5.36}$$

由 (5.35) 可知, 如果 $f(x)$ 是次数 $\leqslant 3$ 的多项式, 则辛普森法给出精确的积分值. 另一方面, 辛普森法是基于被积函数的二次插值, 而这样的插值仅对次数 $\leqslant 2$ 的多项式 $f(x)$ 精确成立. 精度为 3 导致了误差中出现 h^4, 而不是与二次插值误差相应的 h^3. 因为误差中的较高次幂 h^4 和方法的简单表达形式, 所以辛普森法成为最常用的一种数值积分方法.

例 5.2.6 在 (5.19) 中, 我们曾用公式 $S_2(f)$ 计算 $I = \int_0^1 \frac{\mathrm{d}x}{1+x}$. 由

$$f(x) = \frac{1}{1+x}, \quad f^{(3)}(x) = \frac{-6}{(1+x)^4}, \quad f^{(4)}(x) = \frac{24}{(1+x)^5}$$

可知, 其精确的误差为

$$E_n^S(f) = -\frac{h^4}{180} f^{(4)}(c_n), \quad h = \frac{1}{n}$$

其中 $0 \leqslant c_n \leqslant 1$. 所以误差界为

$$\left|E_n^S(f)\right| \leqslant \frac{h^4}{180}(24) = \frac{2h^4}{15}$$

而渐近误差则为

$$\tilde{E}_n^S(f) = -\frac{h^4}{180}\left[\frac{-6}{(1+1)^4} - \frac{-6}{(1+0)^4}\right] = -\frac{h^4}{32}$$

对于 $n=2$, 有 $\tilde{E}^S(f) \doteq -0.001\,95$; 由 (5.20) 知, 实际的误差是 $-0.001\,30$. ■

由 (5.36) 可以推断误差 $I(f) - S_n(f)$ 的性态. 假设 $f'''(b) \neq f'''(a)$, 当 n 翻倍时, h 减半, 而 h^4 减少到原来的 1/16, 所以误差 $E_n^S(f)$ 也减少到原来的 1/16. 这是表 5-2 中看到的 $I^{(1)}$ 和 $I^{(2)}$ 的误差性态. 当 $f'''(b) = f'''(a)$ 时, 误差将减少得更快, 这也部分地解释了为什么表 5-2 中 $I^{(3)}$ 的收敛速度非常快.

倘若

$$\lim_{n\to\infty} \frac{\tilde{E}_n(f)}{E_n(f)} = 1$$

则有渐近误差公式

$$E_n(f) \approx \tilde{E}_n(f) \tag{5.37}$$

使 (5.37) 成立所需的 n 的大小随被积函数 f 而变化, 表 5-2 中的两个例子 $I^{(1)}$ 和 $I^{(2)}$ 说明了这一点. 对于 $I^{(2)}$, 直到 n 增大到 $n \geqslant 64$ 时 (5.37) 才成立.

如果 $f(x)$ 在 $[a,b]$ 上不存在四阶连续的导数, 根据 (5.35) 和 (5.36), 我们同样可以推断辛普森法的收敛效果不会那么好. 这时, 往往需要用其他更有效的数值积分方法.

例 5.2.7 用辛普森法近似计算

$$I = \int_0^1 \sqrt{x}\,\mathrm{d}x = \frac{2}{3}$$

数值结果如表 5-4 所示. “比率” 这一列显示出其收敛速度很慢. ■

表 5-4 $\sqrt{x}$ 的辛普森法

n	误差	比率
2	2.860E−2	
4	1.014E−2	2.82
8	3.587E−3	2.83
16	1.268E−3	2.83
32	4.485E−4	2.83

与梯形法一样, 可以定义修正辛普森法(corrected Simpson's rule) 如下:

$$CS_n(f) = S_n(f) - \frac{h^4}{180}[f'''(b) - f'''(a)] \tag{5.38}$$

这往往是一个比 $S_n(f)$ 更精确的近似积分方法.

5.2.3 理查森外推法

误差估计 (5.32) 和 (5.36) 都具有如下的形式

$$I - I_n \approx \frac{c}{n^p} \tag{5.39}$$

其中 I_n 表示数值积分, h 已被 $(b-a)/n$ 所代替. 常数 c 和 p 的值随着计算方法和被积函数的不同而变化. 对于大多数的被积函数 $f(x)$, 用梯形法时 $p=2$, 用辛普森法时 $p=4$. 对其他一些积分方法, 也有 (5.39) 这样的误差估计. 我们不需要知道 c 的值就可以根据 (5.39) 得到可计算的误差估计 $I - I_n$.

在 (5.39) 中, 用 $2n$ 来代替 n, 得到

$$I - I_{2n} \approx \frac{c}{2^p n^p} \tag{5.40}$$

与 (5.39) 相比, 可以看出

$$2^p[I - I_{2n}] \approx \frac{c}{n^p} \approx I - I_n$$

求解 I 得到

$$\begin{aligned}(2^p - 1)I &\approx 2^p I_{2n} - I_n \\ I &\approx \frac{1}{2^p - 1}[2^p I_{2n} - I_n] \equiv R_{2n}\end{aligned} \tag{5.41}$$

利用 I_n, I_{2n}, p 以及假设 (5.39) 所得到的 R_{2n} 是 I 的一个改进估计, 称为**理查森外推公式**(Richardson's extrapolation formula). 一般来说, R_{2n} 比 I_{2n} 更加精确地逼近 I, 精度的提高程度取决于 (5.39) 和 (5.40) 的有效性.

为了估计 I_{2n} 的误差, 将其与更精确的值 R_{2n} 进行比较

$$I - I_{2n} \approx R_{2n} - I_{2n} = \frac{1}{2^p - 1}[2^p I_{2n} - I_n] - I_{2n}$$

$$I - I_{2n} \approx \frac{1}{2^p - 1}[I_{2n} - I_n] \tag{5.42}$$

这就是**理查森误差估计**(Richardson's error estimate).

例 5.2.8 用梯形法逼近

$$I = \int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x \doteq 0.746\ 824\ 132\ 812\ 43$$

有

$$T_2 \doteq 0.731\ 370\ 251\ 8, \quad T_4 \doteq 0.742\ 984\ 097\ 8$$

取 $p=2$ 和 $n=2$, 根据 (5.41) 式, 得到

$$I \approx R_4 = \frac{1}{3}[4I_4 - I_2] = \frac{1}{3}[4T_4 - T_2] \doteq 0.746\ 855\ 379\ 7$$

R_4 的误差为 $-0.000\ 031\ 2$, 由表 5-1 可知, R_4 比 T_{32} 更精确. 为了估计 T_4 的误差, 根据 (5.42) 式得

$$I - T_4 \approx \frac{1}{3}[T_4 - T_2] \doteq 0.003\ 87$$

T_4 的实际误差为 0.003 84, 所以 (5.42) 是一个非常精确的误差估计. ■

理查森外推和误差估计是很实用的做法, 虽然其效果不一定总能像这个例子那么好. 必须满足的主要假设是 (5.39), 习题 13(b) 给出了检验这个假设对所用的实际值 I_n 是否成立的方法.

大多数计算机程序库中至少含有一种*自动数值积分*(automatic numerical integration) 的计算程序. 这意味着可以根据要求的误差容限, 向用户提供积分值. 程序会在误差允许范围内尝试找到数值积分. 基于多种数值积分方法, 有很多这样的程序. 但它们中的大多数程序使用的误差估计方法都与理查森外推法和估计的思想相关. 由于这些程序结构非常复杂, 我们在这里不作讨论; 但是如果这种积分的数量不是太多, 这些程序一般是求函数积分的最佳方法. 由于积分运算中的各种数据需要大量 "簿记", 这些*程序运行成本*(program overhead costs) 往往很高, 但这样的代价因为使用户节约了编程序的时间而常常是值得的.

5.2.4 周期被积函数

如果

$$f(x) = f(x+\tau), \quad -\infty < x < \infty \tag{5.43}$$

且对于任意比 τ 更小的值, 上式都不成立, 则称函数 $f(x)$ 是以 τ 为周期的*周期函数*. 例如,

$$f(x) = \mathrm{e}^{\cos(\pi x)}$$

是以 $\tau = 2$ 为周期的周期函数. 如果 $f(x)$ 是可微的周期函数, 则它的导数也以 τ 为周期.

考虑用梯形法和辛普森法计算积分

$$I = \int_a^b f(x)\mathrm{d}x$$

并且假设 $b-a$ 是周期 τ 的整数倍, 并且 $f(x)$ 有任意阶的导数. 由 $f(x)$ 的周期性可知, $f(x)$ 的所有的导数都满足

$$f^{(k)}(a) = f^{(k)}(b) \quad k \geqslant 0 \tag{5.44}$$

如果我们考虑梯形法和辛普森法的渐近误差公式, 由 (5.44) 可知这些公式的误差为零. 所以当 $f(x)$ 是光滑周期函数, 且 $b-a$ 是 f 的周期的整数倍时, 误差公式 $E_n^T(f)$ 和 $E_n^S(f)$ 会更迅速地收敛于零.

利用**欧拉–马克劳林展开式**(Euler-MacLaurin expansion), 可把渐近误差公式 $\tilde{E}_n^T(f)$ 和 $\tilde{E}_n^S(f)$ 扩展到 h 的高阶项, 其系数为 $f^{(k)}(b)-f^{(k)}(a)$ 的倍数, 而 $k \geqslant 1$ 是任意奇数. 根据这个事实, 由前面得出的关于周期函数 $f(x)$ 的结论, 可以证明, 误差 $\tilde{E}_n^T(f)$ 和 $\tilde{E}_n^S(f)$ 甚至比推断的结论更迅速地收敛于零. 在处理光滑的周期被积函数时, 梯形法应该作为首选的求积方法. 表 5-1 和表 5-2 中 $I^{(3)}$ 的结果验证了这些结论.

例 5.2.9 边界为

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1$$

的椭圆面积为 πab. 固定面积为 π(即 $ab=1$), 我们来研究当 a 和 b 变化时椭圆周长的变化. 下面给出一些结果而不做详细的讨论, 希望读者自己补充相关细节.

椭圆的参数表示形式为

$$(x,y) = (a\cos\theta, b\sin\theta), \quad 0 \leqslant \theta \leqslant 2\pi \tag{5.45}$$

根据周长计算的标准公式, 以及椭圆关于 x 轴的对称性, 得到椭圆的周长为

$$\begin{aligned} P &= 2\int_0^{\pi} \sqrt{\left(\frac{\mathrm{d}x}{\mathrm{d}\theta}\right)^2 + \left(\frac{\mathrm{d}y}{\mathrm{d}\theta}\right)^2}\mathrm{d}\theta \\ &= 2\int_0^{\pi} \sqrt{a^2\sin^2\theta + b^2\cos^2\theta}\mathrm{d}\theta \end{aligned}$$

因为 $ab=1$, 故上式可以写成

$$\begin{aligned} P(b) &= 2\int_0^{\pi} \sqrt{\frac{1}{b^2}\sin^2\theta + b^2\cos^2\theta}\mathrm{d}\theta \\ &= \frac{2}{b}\int_0^{\pi} \sqrt{(b^4-1)\cos^2\theta + 1}\mathrm{d}\theta \end{aligned} \tag{5.46}$$

我们只考虑 $1 \leqslant b < \infty$ 的情况. 因为两个椭圆

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \quad \text{和} \quad \left(\frac{x}{b}\right)^2 + \left(\frac{y}{a}\right)^2 = 1$$

的周长是相等的,

$$P\left(\frac{1}{b}\right) = P(b), \quad b > 0 \tag{5.47}$$

所以我们只考虑椭圆的 y 轴大于或等于 x 轴的情况. $P(b)$ 的被积函数

$$f(\theta) = \frac{2}{b}\left[(b^4 - 1)\cos^2\theta + 1\right]^{1/2}$$

是以 π 为周期的周期函数. 如上所述, 梯形法是求 (5.46) 的数值积分最合适的方法. 但是随着 b 的变化, 函数 $f(\theta)$ 的性态也会发生变化, 这会给数值积分的精度带来影响. 在图 5-4 中, 我们对几个不同的 b 值给出了对应的函数 $f(\theta)$ 的图像. 在表 5-5 中, 给出了当 n 不断增加时, 对这些 b 值, 利用梯形法的计算结果. 注意到当 b 增加时, 梯形法收敛速度会变慢. 这是因为当 b 增加时, 被积函数 $f(\theta)$ 变化得更快. 对于较大的 b, $f(\theta)$ 在 $\theta = \frac{1}{2}\pi$ 的邻近变化非常快, 与 1 附近那些较小的 b 的情形相比, 这将导致梯形法降低精度. 为了得到一定精度的周长 $P(b)$ 的值, 当 b 增加时我们必须增大 n.

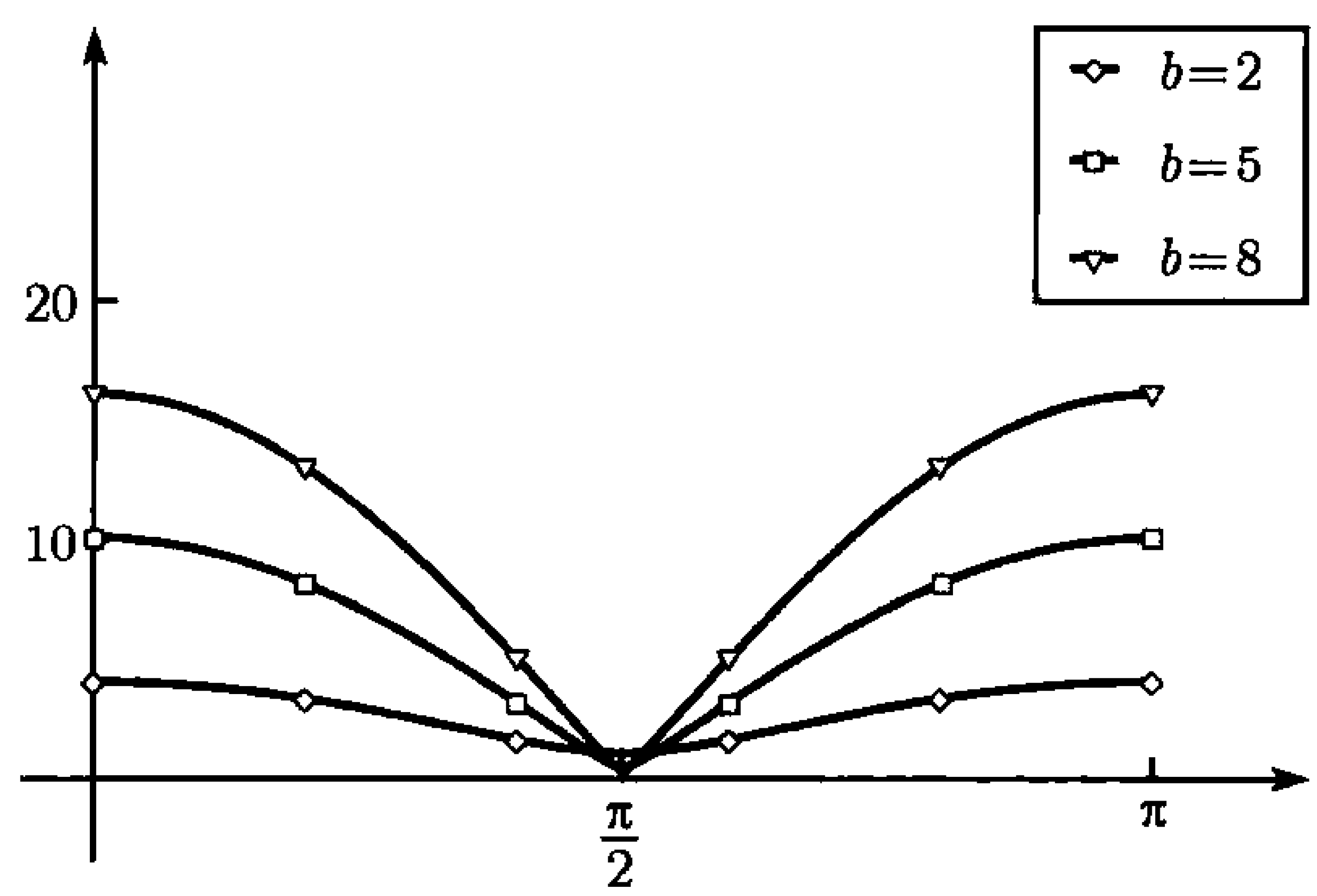


图 5-4 被积函数 $f(\theta)$ 的图像: $b = 2, 5, 8$

表 5-5 (5.46) 的梯形法计算结果

n	$b=2$	$b=5$	$b=8$
8	8.575 517	19.918 814	31.690 628
16	8.578 405	20.044 483	31.953 632
32	8.578 422	20.063 957	32.008 934
64	8.578 422	20.065 672	32.018 564
128	8.578 422	20.065 716	32.019 660
256	8.578 422	20.065 717	32.019 709

图 5-5 中 $P(b)$ 的图像显示了对于较大的 b 有 $P(b) \approx 4b$. 回到 (5.46), 当 b 较

大时，有

$$P(b) \approx \frac{2}{b}\int_0^{\pi}\left[b^4\cos^2\theta\right]^{1/2}\mathrm{d}\theta = \frac{2}{b}b^2\int_0^{\pi}|\cos\theta|\mathrm{d}\theta = 4b$$

我们需要估计上述逼近的误差，以确定何时可以用上式替换 $P(b)$; 在大多数病态情况下，这种方法可以避免求 (5.46) 的积分. 读者可以从几何的角度解释这个方法的合理性. ■

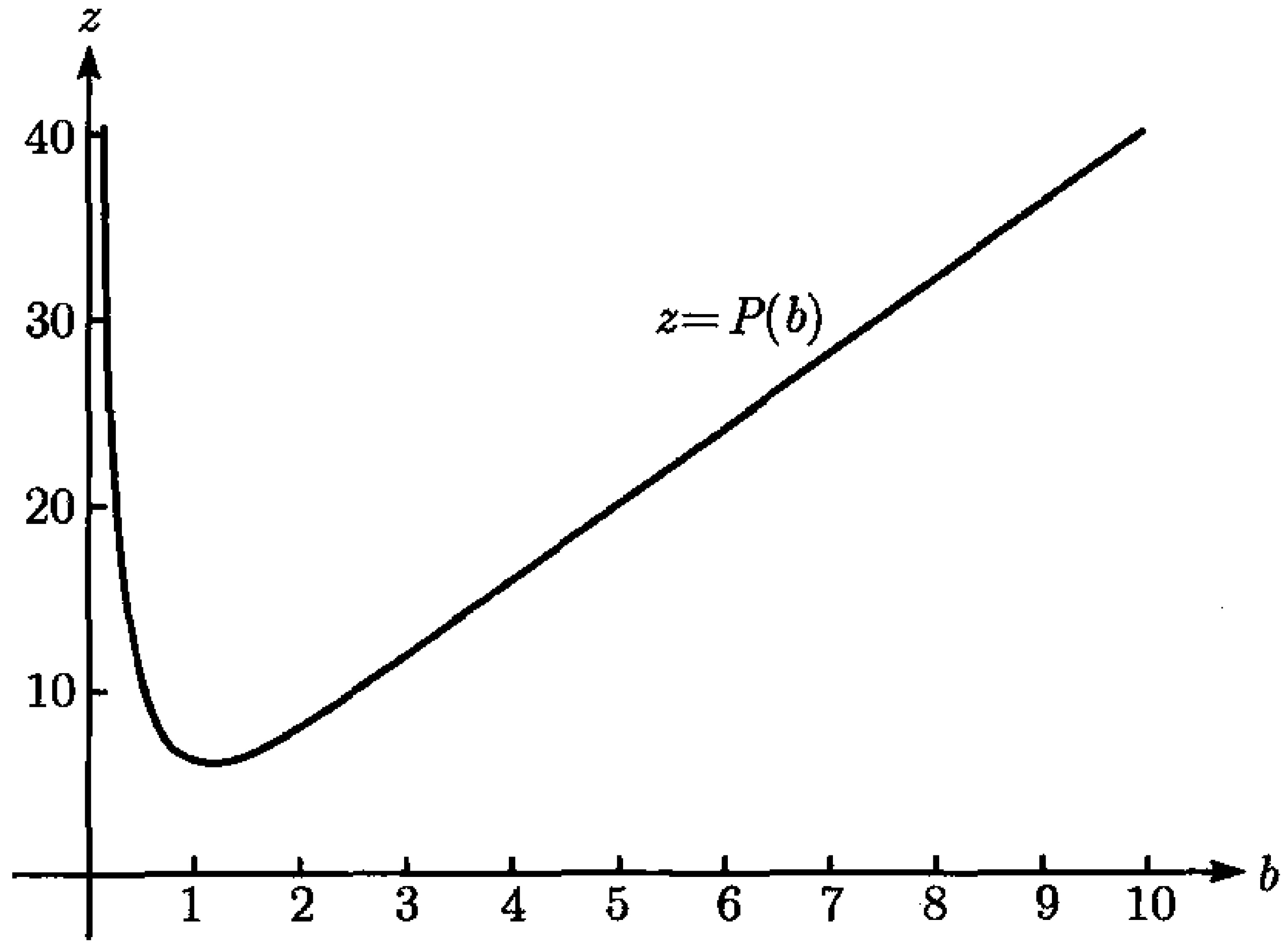


图 5-5 椭圆周长函数 $P(b)$ 的图像

习题

1. 利用误差公式 (5.23), 确定下列积分的估计值 $T_n(f)$ 的误差界.

   (a) $\int_0^{\pi/2}\cos(x)\mathrm{d}x$; **(b)** $\int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x$; **(c)** $\int_0^{\sqrt{\pi}}\cos(x^2)\mathrm{d}x$.

2. 对 5.1 节习题 2 中的积分 (a), (b), (c) 重复习题 1 中的工作. 将其计算出的误差界与实际误差作比较.
3. 把梯形法误差估计公式 (5.32) 应用到 5.1 节习题 2 的积分 (a), (b), (c), (e). 并将结果与其实际误差作比较.
4. 使用 5.1 节表 5-1 的积分 $I^{(2)}$, 重复习题 3 中的工作.
5. 根据梯形法的渐近误差公式 (5.32), 确定子区间个数 n 来计算下列积分, 使其具有给定的精度 ε:

   (a) $\int_1^3 \ln(x)\mathrm{d}x,\quad \varepsilon = 10^{-8}$; **(b)** $\int_0^2 \frac{\mathrm{e}^x-\mathrm{e}^{-x}}{2}\mathrm{d}x,\quad \varepsilon = 10^{-10}$;

   (c) $\int_0^2 \mathrm{e}^{-x^2}\mathrm{d}x,\quad \varepsilon = 10^{-10}$; **(d)** $\int_0^4 \frac{\mathrm{d}x}{1+x^2},\quad \varepsilon = 10^{-12}$.

6. 根据辛普森法和它的渐近误差估计公式 (5.36) 重复习题 5 中的工作.
7. **(a)** 用梯形法 T_n 估计积分

   $$I = \int_1^3 \ln(x)\mathrm{d}x$$

   给出 $I-T_n$ 的严格误差界和渐近误差估计 $I-T_n$. 根据严格误差界, 为了使 $|I-T_n| \leqslant 5\times10^{-8}$, 确定 n 应该是多大.

(b) 用辛普森法重复 (a) 中的工作.

8. 对

$$I = \int_{-1}^{1} \frac{\mathrm{d}x}{2+x}$$

重复习题 7 中的工作.

9. 对

$$I = \int_{-1}^{1} \frac{\mathrm{e}^x + \mathrm{e}^{-x}}{2} \mathrm{d}x$$

重复习题 7 中的工作.

10. 梯形法和辛普森法的误差 $E_n(f)$ 都具有若干有用的性质以帮助简化计算.

(a) 证明: 对于任意的连续函数 $f(x)$ 和 $g(x)$ 都有 $E_n(f+g) = E_n(f) + E_n(g)$.

(b) 证明: 对于任意的连续函数 $f(x)$ 和常数 c 都有 $E_n(cf) = cE_n(f)$.

(c) 如果 $p(x)$ 为线性多项式, 试问 $E_n^T(p)$ 是多少? 如果 $p(x)$ 为三次多项式, 那么 $E_n^S(p)$ 是多少?

(d) 设 $f(x)$ 在 $[a,b]$ 上具有连续的二阶导数, 用线性泰勒多项式

$$p_1(x) = f(a) + (x-a)f'(a)$$

表示 $f(x) = p_1(x) + R_1(x)$, 其中 $R_1(x)$ 为 $p_1(x)$ 的误差. 证明

$$E_n^T(f) = E_n^T(R_1)$$

这个结论有时可以用来简化 $E_n^T(f)$ 的计算过程.

11. 基本的梯形法误差公式 (5.26) 的证明是基于 $f(x)$ 在 α 点的泰勒展开式. 为了证明 (5.26) 式, 将 $f(x)$ 写成

$$f(x) \approx f(\alpha) + (x-\alpha)f'(\alpha) + \frac{(x-\alpha)^2}{2}f''(\alpha)$$

把上式代入 (5.26) 的左端, 从而得到与该式右端十分接近的结果. 可用习题 10(b) 的结论来简化这些计算过程.

12. 回顾公式 (5.30), 它是推导 $E_n^T(f)$ 的误差公式 (5.23) 的中间步骤. 为了完成证明, 应用公式 (A.1) 的思想和附录 A 习题 9 的思想, 并用 $f''(x)$ 代替那些结果中的 $f(x)$.

提示: 将 (5.30) 式方括号中的项写成

$$(b-a)\left[\frac{f''(\gamma_1) + f''(\gamma_2) + \cdots + f''(\gamma_n)}{n}\right]$$

13. (a) 对于所有使得 (5.39) 有效的 n, 由 (5.39) 导出

$$\frac{I - I_n}{I - I_{2n}} \approx 2^p$$

提示: 参考 (5.39) 和 (5.40).

(b) 根据 (5.39) 式, 导出可计算的估计

$$\frac{I_{2n}-I_n}{I_{4n}-I_{2n}}\approx 2^p$$

使用三个相继的值 I_n, I_{2n}, I_{4n}, 这给出了检验 p 值的实用方法. 利用 ln 函数, 得到

$$p=\ln\left(\frac{I_{2n}-I_n}{I_{4n}-I_{2n}}\right)\bigg/\ln 2$$

提示: 记 $I_{2n}-I_n=(I-I_n)-(I-I_{2n})$, 将分母也化成这样的形式, 然后应用 (5.39) 式.

14. (a) 根据习题 13(a) 中的公式以及表 5-4 中的结果, 当辛普森法应用于 $I=\int_0^1\sqrt{x}\mathrm{d}x$ 时, 确定 (5.39) 式中 p 的值.

(b) 把公式 $T_n(f)$ 应用于 $\int_0^1\sqrt{x}\mathrm{d}x$, 其中令 $n=2,4,8,\cdots,128$. 根据习题 13 的 (a) 或 (b), 对这些数值积分计算 (5.39) 式中 p 的值.

15. 下表中的数据是以梯形法计算积分 I 的数值结果, 其中 I 为

$$I=\int_0^1\tan^{-1}x\mathrm{d}x=\frac{1}{4}\pi-\frac{1}{2}\ln 2\doteq 0.438\ 824\ 573\ 117\ 48$$

当 $n=16$, 32, 64 时, 利用下表计算 T_n 的理查森误差估计. 此外, 给出 $n=64$ 时的修正梯形法. 利用上面给出的积分的真实值, 试问用修正梯形法所得的估计值的误差为多少?

n	T_n
4	0.436 206 615 7
8	0.438 172 680 3
16	0.438 661 759 7
32	0.438 783 879 7
64	0.438 814 400 4

16. (a) 下表中的数据是以数值积分公式 I_n 计算积分 I 所得的数值结果, 其中积分真值为 $I=0.3$. 假设存在 $p>0$ 和 c, 使其误差具有下述渐近公式

$$I-I_n\approx\frac{c}{n^p}$$

估计收敛阶 p 的值和 c 的值. 要使得 $|I-I_n|\leqslant 10^{-10}$, 估计 n 应为多大.

n	I_n	n	I_n
8	0.299 333 176 5	64	0.299 979 155 6
16	0.299 789 913 9	128	0.299 993 434 4
32	0.299 933 823 9	256	0.299 997 932 0

(b) 假设 I 是未知的 (积分的真值往往无法得知), 估计 p.

17. 用理查森外推法估计 5.1 节中习题 2 的 (a), (b), (c) 和 (e) 中的误差.

18. 用理查森外推法估计 5.1 节中习题 3 的 (a), (b), (c) 和 (e) 中的误差.

19. 下表中给出了数值积分的值及它们的差. 假设误差满足 $I - I_n \approx c/n^p$, 求出 p 适当的值. 并估计 I_{64} 的误差.

提示: 利用习题 13(b).

n	I_n	$I_n - I_{n/2}$
2	0.702 877 396	
4	0.781 978 959	0.079 10
8	0.804 500 932	0.022 52
16	0.810 303 086	0.005 802
32	0.811 764 354	0.001 461
64	0.812 130 341	0.000 366 0

20. (a) 对 $I = \int_0^1 \sin(\sqrt{x})\mathrm{d}x$ 用辛普森法, 取 $n = 2, 4, 8, \cdots, 128$. 根据习题 13(b) 的公式计算收敛速度.

(b) 使用变量代换 $x = t^2$ 对积分 I 进行变换, 得到

$$I = 2\int_0^1 t\sin(t)\mathrm{d}t$$

对这个新的积分用辛普森法, 取 $n = 2, 4, 8, \cdots, 128$, 并与 (a) 的结果作比较.

21. (a) 用辛普森法求 $I = \int_0^1 \sin(\sqrt[3]{x})\mathrm{d}x$, 其中取 $n = 2, 4, 8, \cdots, 128$. 根据习题 13(b) 的公式计算收敛速度.

(b) 根据习题 20(b) 中的思想, 对积分作合适的变量代换, 得到一个新的被积函数, 使得辛普森法对新的积分具有较高的收敛阶.

22. 在只知道近似的节点值 $f(x_i)$ 的情况下考虑 $I = \int_a^b f(x)\mathrm{d}x$ 的数值积分. 更具体地说, 令 $\hat{f}_i \approx f(x_i)$, $i = 0, 1, \cdots, n$, 并假设

$$\left|f(x_i) - \hat{f}_i\right| \leqslant \varepsilon, \quad i = 0, 1, \cdots, n$$

设 I_n 和 $\hat{I}_n$ 是分别利用 $\{f(x_i)\}$ 和 $\{\hat{f}_i\}$ 所求得的数值积分.

(a) 当 I_n 是利用梯形法或利用辛普森法求得时, 证明

$$\left|I_n - \hat{I}_n\right| \leqslant \varepsilon(b - a)$$

(b) 将这一结果推广到所有满足下列条件的求积公式: (i) 所有的积分的权大于零, (ii) 积分公式的精度大于等于零.

5.3 高斯数值积分

本章前两节中学习的数值方法都是基于对线性插值多项式和二次插值多项式求积, 得到的求积公式可以用于不断细分的子区间. 本节中, 我们考虑能对次数不

断增高的多项式精确积分的数值方法, 而且不用对积分区间再作细分. 为此, 我们先回顾 4.4 节中函数逼近的有关内容.

设 $f(x)$ 在 $[a,b]$ 上连续. 在 $a \leqslant x \leqslant b$ 上用一个次数 $\leqslant n$ 的多项式 $p(x)$ 去逼近 $f(x)$, 并用 $\rho_n(f)$ 表示这种逼近能达到的最小误差界. 达到最小误差界的逼近多项式 $m_n(x)$ 称为 $f(x)$ 的 n 次极小极大逼近, 则有

$$\max_{a\leqslant x\leqslant b} |f(x) - m_n(x)| = \rho_n(f) \tag{5.48}$$

$\rho_n(f)$ 称为极小极大误差. 由 (4.82) 式知, $\rho_n(f)$ 收敛于零的速度往往非常快.

例 5.3.1 令 $f(x) = \mathrm{e}^{-x^2}$, 其中 $0 \leqslant x \leqslant 1$. 对于 $n = 1, 2, \cdots, 10$, 表 5-6 给出了极小极大误差 $\rho_n(f)$. 尽管收敛速度不一致, 但它们收敛于零的速度很快. ■

表 5-6 $\mathrm{e}^{-x^2}(0 \leqslant x \leqslant 1)$ 的极小极大误差

n	$\rho_n(f)$	n	$\rho_n(f)$
1	5.30E−2	6	7.82E−6
2	1.79E−2	7	4.62E−7
3	6.63E−4	8	9.64E−8
4	4.63E−4	9	8.05E−9
5	1.62E−5	10	9.16E−10

如果我们有一种数值积分公式可以对一定次数的多项式进行精确积分, 则当函数 $f(x)$ 可以很好地用这些多项式逼近时, 我们相信上述公式能有效地计算函数 $f(x)$ 的积分值. 为了说明这类积分公式的由来, 我们不妨考虑积分

$$I(f) = \int_{-1}^{1} f(x)\mathrm{d}x \tag{5.49}$$

积分 (5.49) 与一般区间 $[a,b]$ 上积分之间的关系将在后文讨论.

积分公式具有如下的一般形式

$$I_n(f) = \sum_{j=1}^{n} w_j f(x_j) \tag{5.50}$$

我们需要选择节点 $\{x_1, \cdots, x_n\}$ 和权 $\{w_1, \cdots, w_n\}$, 使得对次数尽可能高的所有多项式 $f(x)$, 都有 $I_n(f) = I(f)$.

$n = 1$ 的情形 求积公式的形式如下

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx w_1 f(x_1) \tag{5.51}$$

它对次数尽可能高的多项式都精确成立.

取 $f(x) \equiv 1$ 并且令 (5.51) 式为等式, 得到

$$2 = w_1$$

取 $f(x) = x$ 并且再次令 (5.51) 式为等式, 则

$$0 = w_1 x_1$$

由此可知 $x_1 = 0$. 所以 (5.51) 式变为

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx 2f(0) \equiv I_1(f) \tag{5.52}$$

这是 5.1 节习题 8(a) 中的中点法公式. (5.22) 式对所有线性多项式都精确成立, 其证明过程作为习题 7 留给读者完成.

为了证明 (5.52) 式对二次多项式不能精确成立, 我们取 $f(x) = x^2$. 则公式 (5.52) 的误差为

$$\int_{-1}^{1} x^2 \mathrm{d}x - 2 \times (0)^2 = \frac{2}{3} \neq 0$$

根据 5.1 节习题 11 的结果知, (5.52) 的精度为 1.

$n = 2$ 的情形 积分公式为

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx w_1 f(x_1) + w_2 f(x_2) \tag{5.53}$$

它有 4 个未知数: x_1, x_2, w_1, w_2. 为了确定这些参数, 我们要求求积公式能精确计算下列 4 个单项式的积分值

$$f(x) = 1, x, x^2, x^3 \tag{5.54}$$

因而得到下列四个方程

$$\begin{aligned} 2 &= w_1 + w_2 \\ 0 &= w_1 x_1 + w_2 x_2 \\ \frac{2}{3} &= w_1 x_1^2 + w_2 x_2^2 \\ 0 &= w_1 x_1^3 + w_2 x_2^3 \end{aligned} \tag{5.55}$$

这是一个四元非线性方程组, 可以证明它的解是

$$w_1 = w_2 = 1, \quad x_1 = -\frac{\sqrt{3}}{3}, \quad x_2 = \frac{\sqrt{3}}{3} \tag{5.56}$$

互换 x_1 和 x_2 的符号后可得到方程组的另一个解.

这就得到积分公式

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) \equiv I_2(f) \tag{5.57}$$

由于积分公式对于 (5.54) 中的单项式的积分都精确, 所以它能精确积分所有次数 $\leqslant 3$ 的多项式 (参见习题 6). 通过直接计算也可以说明求积公式不能精确计算 4 次多项式 $f(x)=x^4$ 的积分. 因此, $I_2(f)$ 的精度为 3.

例 5.3.2 计算

$$I = \int_{-1}^{1} \mathrm{e}^x \mathrm{d}x = \mathrm{e} - \mathrm{e}^{-1} \doteq 2.350\,402\,4$$

的近似值. 根据 (5.57), 得到

$$I_2 = \mathrm{e}^{-\sqrt{3}/3} + \mathrm{e}^{\sqrt{3}/3} \doteq 2.342\,696\,1$$

$$I - I_2 \doteq 0.007\,71$$

仅使用两个节点的积分近似值的误差已非常小. ■

$n>2$ 的情形 我们探讨含有 $2n$ 个未知数 $x_1,\cdots,x_n,w_1,\cdots,w_n$ 的公式 (5.50). 要求积分公式对于下列 $2n$ 个单项式都精确成立

$$f(x) = 1, x, x^2, \cdots, x^{2n-1} \tag{5.58}$$

也就是说, 对所有次数 $\leqslant 2n-1$ 的多项式 f, 要求等式 $I_n(f)=I(f)$ 都精确成立. 由此得到下列关于 $2n$ 个未知数和 $2n$ 个非线性方程的方程组:

$$\begin{aligned}
2 &= w_1 + w_2 + \cdots + w_n \\
0 &= w_1x_1 + w_2x_2 + \cdots + w_nx_n \\
\frac{2}{3} &= w_1x_1^2 + w_2x_2^2 + \cdots + w_nx_n^2 \\
0 &= w_1x_1^3 + w_2x_2^3 + \cdots + w_nx_n^3 \\
&\ \ \vdots \\
\frac{2}{2n-1} &= w_1x_1^{2n-2} + \cdots + w_nx_n^{2n-2} \\
0 &= w_2x_1^{2n-1} + \cdots + w_nx_n^{2n-1}
\end{aligned} \tag{5.59}$$

最终求得的积分公式 $I_n(f)$ 的精度为 $2n-1$.

求解这样的非线性方程组非常困难. 好在几个最常用的 n 所对应的节点 $\{x_i\}$ 和权 $\{w_i\}$ 的值已被计算出来并收集在表格中. 表 5-7 给出了 $n=2,3,\cdots,8$ 所对

应的值. 更完整的表格数据可以参见 A. Stround and D. Secrest(1966). 大多数计算机中心都有用来计算节点和权的程序, 或者直接运行数值积分的程序.

表 5-7 高斯求积公式的节点及其权

n	x_i	w_i
2	±0.577 350 269 2	1.0
3	±0.774 596 669 2	0.555 555 555 6
	0.0	0.888 888 888 9
4	±0.861 136 311 6	0.347 854 845 1
	±0.339 981 043 6	0.652 145 154 9
5	±0.906 179 845 9	0.236 926 885 1
	±0.538 469 310 1	0.478 628 670 5
	0.0	0.568 888 888 9
6	±0.932 469 514 2	0.171 324 492 4
	±0.661 209 386 5	0.360 761 573 0
	±0.238 619 186 1	0.467 913 934 6
7	±0.949 107 912 3	0.129 484 966 2
	±0.741 531 185 6	0.279 705 391 5
	±0.405 845 151 4	0.381 830 050 5
	0.0	0.417 959 183 7
8	±0.960 289 856 5	0.101 228 536 3
	±0.796 666 477 4	0.222 381 034 5
	±0.525 532 409 9	0.313 706 645 9
	±0.183 434 642 5	0.362 683 783 4

还可以利用**正交多项式理论**(theory of orthogonal polynomials), 来讨论数值积分公式 (5.50). 根据上述理论, 可证明节点 $\{x_1,\cdots,x_n\}$ 是 $[-1,1]$ 上 n 次**勒让德多项式**(Legendre polynomial) 的零点. 4.7 节中曾介绍过这些多项式. 例如

$$P_2(x)=\frac{1}{2}(3x^2-1)$$

它的根就是 (5.56) 中给出的节点. 因为勒让德多项式是众所周知的, 所以不需要借助非线性方程组 (5.59) 就可以找出节点 $\{x_j\}$. 关于这个理论的介绍, 可参见 Atkinson(1989, p. 270).

公式序列 (5.50) 称为**高斯数值积分**(Gaussian numerical integration) 方法. 由定义知, $I_n(f)$ 选用 n 个节点, 它对所有次数 $\leqslant 2n-1$ 的多项式都精确. 由 (5.49) 知 $I_n(f)$ 用于计算区间 $[-1,1]$ 上的积分, 这个限制很容易被取消. 对一般区间 $[a,b]$ 上的积分

$$I(f)=\int_a^b f(x)\mathrm{d}x \tag{5.60}$$

引进线性的变量替换

$$x = \frac{b+a+t(b-a)}{2}, \quad -1 \leqslant t \leqslant 1 \tag{5.61}$$

可将积分化为

$$I(f) = \frac{b-a}{2} \int_{-1}^{1} \tilde{f}(t)\mathrm{d}t \tag{5.62}$$

其中

$$\tilde{f}(t) = f\left(\frac{b+a+t(b-a)}{2}\right) \tag{5.63}$$

然后, 可应用公式 $I_n(\tilde{f})$ 计算这个新的积分.

例 5.3.3 我们曾在 5.1 节中举例用梯形法和辛普森法计算了 (5.14)~(5.16) 定义的积分 $I^{(1)}$, $I^{(2)}$ 和 $I^{(3)}$. 现在用高斯数值积分重新计算这些积分. 先按照上述方法, 把它们化为区间 $[-1,1]$ 上的积分. 误差结果如表 5-8 所示, 表中 (*) 表示相对于计算机使用的精度 (大约 16 位十进制数字), 其误差为零.

如果将这些误差与表 5-1 和表 5-2 中的误差进行比较, 则可以发现 $I^{(1)}$ 和 $I^{(2)}$ 的高斯积分比梯形法和辛普森法更有效. 但在计算周期被积函数的积分 $I^{(3)}$ 时, 高斯积分法没有梯形法或辛普森法有效. 除了周期被积函数, 高斯数值积分方法往往比梯形法和辛普森法更精确. 这个结论对大多数积分, 甚至被积函数没有连续导数的积分也都是成立的. ■

表 5-8 高斯数值积分的实例

n	$I^{(1)}$ 的误差	$I^{(2)}$ 的误差	$I^{(3)}$ 的误差
2	2.29E−4	−2.33E−2	8.23E−1
3	9.55E−6	−3.49E−2	−4.30E−1
4	−3.35E−7	−1.90E−3	1.77E−1
5	6.05E−9	1.70E−3	−8.12E−2
6	−7.77E−11	2.74E−4	3.55E−2
7	7.89E−13	−6.45E−5	−1.58E−2
10	*	1.27E−6	1.37E−3
15	*	7.40E−10	−2.33E−5
20	*	*	3.96E−7

表 5-9 (5.64) 的高斯积分

n	$I-I_n$	比率
2	−7.22E−3	
4	−1.16E−3	6.2
8	−1.69E−4	6.9
16	−2.30E−5	7.4
32	−3.00E−6	7.6
64	−3.84E−7	7.8

例 5.3.4　用高斯积分方法计算

$$I=\int_0^1 \sqrt{x}\mathrm{d}x=\frac{2}{3} \tag{5.64}$$

计算结果如表 5-9 所示, n 表示节点个数. “比率” 这一列的计算公式为

$$\frac{I-I_{\frac{1}{2}n}}{I-I_n}$$

由此可见, 对某个常数 c, 误差近似地满足

$$I-I_n\approx\frac{c}{n^3} \tag{5.65}$$

由辛普森法计算积分 (5.64) 的结果列于表 5-4, 把 (5.65) 与表 5-4 进行比较. 经验上的收敛速度与 $1/n^{1.5}$ 成比例, 这比 (5.65) 给出的高斯积分法的收敛速度慢了很多. ■

我们给出另外一个将最小最大误差与高斯数值积分误差联系起来的结果.

定理 5.3.5　设 $f(x)$ 在 $a\leqslant x\leqslant b$ 上连续且 $n\geqslant 1$. 若我们用高斯数值积分方法计算 $I(f)=\int_a^b f(x)\mathrm{d}x$, 则 I_n 的误差满足

$$|I(f)-I_n(f)|\leqslant 2(b-a)\rho_{2n-1}(f) \tag{5.66}$$

其中 $\rho_{2n-1}(f)$ 表示区间 $[a,b]$ 上 $f(x)$ 的 $2n-1$ 次极小极大误差.

例 5.3.6　根据表 5-6, 使用公式 (5.66) 计算积分

$$I=\int_0^1 \mathrm{e}^{-x^2}\mathrm{d}x \tag{5.67}$$

当 $n=3$ 时, 上述误差界给出

$$|I-I_3|\leqslant 2\rho_5(\mathrm{e}^{-x^2})\doteq 3.24\times 10^{-5}$$

由表 5-8 可得, 它的实际误差为 9.55E − 6. ■

高斯数值积分方法使用起来并不像梯形法和辛普森法那样简单, 部分原因在于高斯求积法没有简单的公式来计算节点及权, 而且较难估计其误差. 然而, 由于高斯求积法的收敛速度非常快, 所以当人们需要计算许多积分时, 往往考虑使用高斯求积法. 鉴于误差估计非常困难, 所以在大多数情况下我们观察两个或许多个相继的值. 若 n 翻倍, 则反复比较两个相继值 I_n 和 I_{2n}, 就可以较好地估计出 I_n 的误差

$$I-I_n\approx I_{2n}-I_n \tag{5.68}$$

虽然这种方法使用起来不是很有效, 但因为 I_n 的收敛速度非常快, 所以这个方法还是可取的.

加权高斯求积

计算下列形式的积分

$$I(f)=\int_a^b w(x)f(x)\mathrm{d}x \tag{5.69}$$

是一个常见的问题, 其中 $f(x)$ 为 "良态" 的函数, 而 $w(x)$ 可能是 (甚至经常是) 病态的函数. 对于许多函数 $w(x)$, 可以推广高斯积分法来处理这样的积分. 例如

$$\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}}\mathrm{d}x,\quad \int_0^1 \sqrt{x}f(x)\mathrm{d}x,\quad \int_0^1 f(x)\ln\left(\frac{1}{x}\right)\mathrm{d}x$$

函数 $w(x)$ 称为**权函数**(weight function).

我们仿效本节开始时给出的讨论方法处理下面特殊情况的积分:

$$I(f)=\int_0^1 \frac{f(x)}{\sqrt{x}}\mathrm{d}x \tag{5.70}$$

其中 $w(x)=1/\sqrt{x}$. 和前面一样, 我们探讨如下形式的数值积分公式:

$$I_n(f)=\sum_{j=1}^n w_j f(x_j) \tag{5.71}$$

要求选择节点 $\{x_1,\cdots,x_n\}$ 和权 $\{w_1,\cdots,w_n\}$, 使得对次数尽可能高的多项式 $f(x)$, $I_n(f)=I(f)$ 都成立.

$n=1$ 的情形 积分公式的形式为

$$\int_0^1 \frac{f(x)}{\sqrt{x}}\mathrm{d}x \approx w_1 f(x_1)$$

取 $f(x)=1$ 和 $f(x)=x$, 使上式精确成立, 则得到两个方程

$$w_1=\int_0^1 \frac{1}{\sqrt{x}}\mathrm{d}x=2$$

$$w_1x_1=\int_0^1 \frac{x}{\sqrt{x}}\mathrm{d}x=\frac{2}{3}$$

解出 w_1 和 x_1, 得到公式

$$\int_0^1 \frac{f(x)}{\sqrt{x}}\mathrm{d}x \approx 2f\left(\frac{1}{3}\right) \tag{5.72}$$

它的精度为 1.

$n=2$ 的情形 积分公式的形式为

$$\int_0^1 \frac{f(x)}{\sqrt{x}}\mathrm{d}x \approx w_1f(x_1)+w_2f(x_2) \tag{5.73}$$

取 $f(x)=1,x,x^2,x^3$, 使上式精确成立, 则得到四个方程

$$\begin{aligned}
w_1+w_2&=\int_0^1\frac{1}{\sqrt{x}}\mathrm{d}x=2\\
w_1x_1+w_2x_2&=\int_0^1\frac{x}{\sqrt{x}}\mathrm{d}x=\frac{2}{3}\\
w_1x_1^2+w_2x_2^2&=\int_0^1\frac{x^2}{\sqrt{x}}\mathrm{d}x=\frac{2}{5}\\
w_1x_1^3+w_2x_2^3&=\int_0^1\frac{x^3}{\sqrt{x}}\mathrm{d}x=\frac{2}{7}
\end{aligned}$$

方程组的解是

$$\begin{aligned}
x_1&=\frac{3}{7}-\frac{2}{35}\sqrt{30}\doteq 0.115\ 59,\quad x_2=\frac{3}{7}+\frac{2}{35}\sqrt{30}\doteq 0.741\ 56\\
w_1&=1+\frac{1}{18}\sqrt{30}\doteq 1.304\ 29,\quad w_2=1-\frac{1}{18}\sqrt{30}\doteq 0.695\ 71
\end{aligned}$$

最终求得的公式 (5.73) 的精度为 3.

$n>2$ 的情形 使用含有 $2n$ 个未知数 $x_1,\cdots,x_n,w_1,\cdots,w_n$ 的公式 (5.71). 迫使积分公式对下列 $2n$ 个单项式都精确成立

$$f(x)=1,x,x^2,\cdots,x^{2n-1}$$

也就是说, 迫使 $I_n(f)=I(f)$ 对所有次数 $\leqslant 2n-1$ 的多项式 f 都精确成立. 从而导致下列含有 $2n$ 个未知数和 $2n$ 个非线性方程的方程组:

$$\begin{aligned}
w_1+w_2+\cdots+w_n&=2\\
w_1x_1+w_2x_2+\cdots+w_nx_n&=\frac{2}{3}\\
w_1x_1^2+w_2x_2^2+\cdots+w_nx_n^2&=\frac{2}{5}\\
&\vdots\\
w_1x_1^{2n-1}+\cdots+w_nx_n^{2n-1}&=\frac{2}{4n-1}
\end{aligned}\tag{5.74}$$

最终求得的公式 $I_n(f)$ 的精度为 $2n-1$. 与前面一样, 直接求解这个方程组是十分困难的, 但是可以通过另外的途径来得到 $\{x_i\}$ 和 $\{w_i\}$. 这个方法是基于考察与区间 $[0,1]$ 上的权函数

$$w(x)=\frac{1}{\sqrt{x}}$$

相应的正交多项式来实现的.

例 5.3.7 计算

$$I=\int_0^1 \frac{\cos(\pi x)}{\sqrt{x}}\mathrm{d}x \doteq 0.747\,965\,666\,831\,46$$

使用 (5.72) 和 (5.73), 得到

$$I_1 = 1.0$$
$$I_2 \doteq 0.740\,519$$

I_2 是 I 的一个较合理的估计值, 误差 $I - I_2 \doteq 0.007\,45$. ■

还可以讨论如下加权高斯积分法的一般理论

$$I(f)=\int_a^b w(x)f(x)\mathrm{d}x \approx \sum_{j=1}^{n} w_j f(x_j) = I_n(f) \tag{5.75}$$

要求权函数 $w(x)$ 满足下列条件:

A1. 对 $a<x<b$, $w(x)>0$;

A2. 对所有的整数 $n\geqslant 0$, 都有

$$\int_a^b w(x)\,|x|^n \mathrm{d}x < \infty$$

这两个假设与 4.7 节广义最小二乘逼近理论中 (4.124) 式后面的假设相同. 这并不是巧合, 原因是高斯求积法和最小二乘逼近理论都与正交多项式相关联. 求解 (5.74) 得到的节点 $\{x_i\}$ 是权函数 $w(x)=1/\sqrt{x}$ 在区间 $[a,b]$ 上 n 次正交多项式的零点. 对于一般形式 (5.75), 其节点 $\{x_i\}$ 是权函数 $w(x)$ 在区间 $[a,b]$ 上 n 次正交多项式的零点. 关于一般高斯积分法 (5.75) 更多的内容, 请参见 Atkinson(1989, 5.3 节).

习题

1. 回顾 (5.57) 式后面的例子, 用公式 I_3 和 I_4 计算 $\int_{-1}^{1} \mathrm{e}^x \mathrm{d}x$ 的值. 使用表 5-7 给出的节点和系数.
2. 利用公式 I_2, I_3 和 I_4 计算 5.1 节习题 2 的 (a)~(f) 中的积分. 计算出误差并将其与前面梯形法和辛普森法的误差进行比较. 如果有现成的高斯数值积分方法程序, 则对 $I_5,\cdots,I_{10}$ 做同样的计算.
3. 对于 $n=2,4,5$, 重复 (5.67) 式后面的例子.
4. 对于 $n=1,2,\cdots,8$, 用高斯数值积分方法, 重复 5.1 节习题 4 中的工作. 如 (5.60)~(5.63) 所示, 先将积分化为标准区间 $[-1,1]$ 上的积分.
5. 对于 $n=1,2,\cdots,8$, 用高斯数值积分方法, 重复 5.1 节习题 5 中的工作. 按照 (5.60)~(5.63), 先将积分化为标准区间 $[-1,1]$ 上的积分.

6. 证明：如果积分公式 (5.50) 对 $1, x, x^2, \cdots, x^m$ 精确成立，则积分公式对所有次数 $\leqslant m$ 的多项式也都精确成立.

提示：参考 5.2 节习题 10.

7. 证明：单点公式 (5.52) 对任意的线性多项式 $f(x)$ 都精确成立.

8. 确定公式

$$\int_0^1 f(x)\mathrm{d}x \approx c_1 f(0) + c_2 f(1)$$

中的常数 c_1 和 c_2，使得上式对次数尽可能高的所有多项式都精确成立. 求出其精度.

9. 确定公式

$$\int_0^1 f(x)\mathrm{d}x \approx w_1 f(0) + w_2 f(x_2)$$

的权 w_1、权 w_2 和节点 x_2，使得公式对次数尽可能高的所有多项式都精确成立. 给出其精度.

10. 考虑积分

$$I(f) = \int_0^1 f(x) \ln\left(\frac{1}{x}\right)\mathrm{d}x$$

其中函数 $f(x)$ 在 $0 \leqslant x \leqslant 1$ 上有多阶连续的导数. 重复 (5.69)~(5.75) 的做法，讨论下列公式.

(a) 确定公式

$$\int_0^1 f(x) \ln\left(\frac{1}{x}\right)\mathrm{d}x \approx w_1 f(x_1) \equiv I_1(f)$$

使其对任意线性多项式 $f(x)$ 都精确成立.

提示：$\ln\left(\frac{1}{x}\right) = -\ln x$，且

$$\int_0^1 x^m \ln(x)\mathrm{d}x = \frac{-1}{(m+1)^2}, \quad m \geqslant 0$$

(b) 确定公式

$$\int_0^1 f(x) \ln\left(\frac{1}{x}\right)\mathrm{d}x \approx w_1 f(x_1) + w_2 f(x_2) \equiv I_2(f)$$

使其对所有次数 $\leqslant 3$ 的多项式都精确成立，建立含有未知数 w_1, w_2, x_1, x_2 的四个方程的方程组. 虽然可以用 5.3 节中的方法来求解这个方程组，但这里我们只要求验证

$$x_1 = \frac{15 - \sqrt{106}}{42}, \quad x_2 = \frac{15 + \sqrt{106}}{42}$$

$$w_1 = \frac{21}{\sqrt{106}}\left(x_2 - \frac{1}{4}\right), \quad w_2 = 1 - w_1$$

是方程组的解.

(c) 利用 (a) 中的公式 I_1 和 (b) 中的公式 I_2 计算

$$I = \int_0^1 \cos(x) \ln\left(\frac{1}{x}\right)\mathrm{d}x$$

它的真值为 $I \doteq 0.946\ 083\ 1$.

(d) 对于 $n=2,4,8$, 利用中点法 [参见 5.1 节习题 8(b)] 计算 I. 并与 (c) 的结果进行比较.

11. 考虑如下形式的积分

$$I(f)=\int_0^1 \sqrt{x}f(x)\mathrm{d}x$$

其中 $f(x)$ 在 $[0,1]$ 上有多阶连续的导数.

(a) 确定公式

$$\int_0^1 \sqrt{x}f(x)\mathrm{d}x \approx w_1 f(x_1) \equiv I_1(f)$$

使其对任意的线性多项式 $f(x)$ 精确成立.

(b) 确定公式

$$\int_0^1 \sqrt{x}f(x)\mathrm{d}x \approx w_1 f(x_1)+w_2 f(x_2) \equiv I_2(f)$$

使其对所有次数 $\leqslant 3$ 的多项式都精确成立, 建立含有未知数 w_1,w_2,x_1,x_2 的四个方程的方程组. 验证

$$x_1=\frac{1}{9}\left(5+2\sqrt{\frac{10}{7}}\right),\quad x_2=\frac{1}{9}\left(5-2\sqrt{\frac{10}{7}}\right),$$
$$w_1=\frac{1}{15}\left(5+\sqrt{\frac{7}{10}}\right),\quad w_2=\frac{2}{3}-w_1$$

是方程组的解. 可以用数值方法 (例如利用 MATLAB) 或符号方法 (例如利用 Maple 或 Mathematica) 验证.

(c) 使用公式 I_1 和 I_2 计算:

$$I=\int_0^1 \sqrt{x}\mathrm{e}^{-x}\mathrm{d}x \doteq 0.378\ 944\ 691\ 64$$

(d) 对于 $n=2,4,8$, 利用中点法 [参见 5.1 节习题 8(b)] 计算 I. 并与 (c) 的结果进行比较.

12. 证明定理 5.3.5. 设 $f(x)$ 是 $[a,b]$ 上的连续函数. 对 $n\geqslant 0$, 设 $q_{2n-1}(x)$ 是 $f(x)$ 的 $2n-1$ 次多项式极小极大逼近:

$$\max_{a\leqslant x\leqslant b}|f(x)-q_{2n-1}(x)|=\rho_{2n-1}(f)$$

(a) 证明 $I_n(q_{2n-1})=I(q_{2n-1})$

(b) 证明

$$I(f)-I_n(f)=I(f-q_{2n-1})-I_n(f-q_{2n-1})$$

(c) 当 $n\geqslant 1$ 时, 可以证明高斯求积公式 $I_n(f)$ 中的权都满足 $w_i>0$. 试证明

$$|I(f)-I(q_{2n-1})|\leqslant (b-a)\rho_{2n-1}(f)$$
$$|I_n(f)-I_n(q_{2n-1})|\leqslant (b-a)\rho_{2n-1}(f)$$

然后完成定理 5.3.5 的证明.

5.4 数值微分

为了数值计算 $f(x)$ 的导数, 先回顾导数的定义

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

当 h 的值很小时, 由上式可得

$$f'(x)\approx\frac{f(x+h)-f(x)}{h}\equiv \mathrm{D}_hf(x) \tag{5.76}$$

$\mathrm{D}_hf(x)$ 是 $f(x)$ 以 h 为步长的**数值导数**(numerical derivative).

例 5.4.1 利用 $\mathrm{D}_hf(x)$ 逼近 $f(x)=\cos(x)$ 在 $x=\pi/6$ 处的导数. 表 5-10 给出了不同 h 值的数值结果. 观察 "误差" 这一列, 可发现误差几乎与 h 成正比; 当 h 减半时, 误差也几乎减半. ■

表 5-10 用公式 (5.76) 计算 $f(x)=\cos(x)$ 的数值微分

h	D_hf	误差	比率
0.1	−0.542 43	0.042 43	
0.05	−0.521 44	0.021 44	1.98
0.025	−0.510 77	0.010 77	1.99
0.012 5	−0.505 40	0.005 40	1.99
0.006 25	−0.502 70	0.002 70	2.00
0.003 125	−0.501 35	0.001 35	2.00

为了解释此例中误差的性态, 我们根据泰勒定理导出一个误差公式. 将函数 $f(x+h)$ 在 x 点处展开, 得到

$$f(x+h)=f(x)+hf'(x)+\frac{h^2}{2}f''(c)$$

其中 c 为 x 和 $x+h$ 之间的某个数. 把上式代入 (5.76) 式右端, 得到

$$\begin{aligned}\mathrm{D}_hf(x)&=\frac{1}{h}\left\{\left[f(x)+hf'(x)+\frac{h^2}{2}f''(c)\right]-f(x)\right\}\\&=f'(x)+\frac{h}{2}f''(c)\end{aligned}$$

$$f'(x)-\mathrm{D}_hf(x)=-\frac{h}{2}f''(c) \tag{5.77}$$

误差基本上与 h 成正比, 这与表 5-10 中的结果相符. 在上例中,

$$f'\left(\frac{\pi}{6}\right)-\mathrm{D}_hf\left(\frac{\pi}{6}\right)=\frac{h}{2}\cos(c) \tag{5.78}$$

其中 c 在 $\frac{1}{6}\pi$ 和 $\frac{1}{6}\pi+h$ 之间. 若 c 取 $\frac{1}{6}\pi$, 则 (5.78) 式右端与表 5-10 中的误差列的值大致吻合, 读者可以自行验证这一点.

如例 5.4.1 所示, 在公式 (5.76) 中, 步长是正数, 即 $h>0$. 公式 (5.76) 一般称为一阶导数的**向前差分公式**(forward difference formula). 可以把 (5.76) 中的 h 改为 $-h$, 得到公式

$$f'(x)\approx\frac{f(x)-f(x-h)}{h},\quad h>0 \tag{5.79}$$

这是一阶导数的**向后差分公式**(backward difference formula). 类似于公式 (5.77) 的推导, 可得出

$$f'(x)-\frac{f(x)-f(x-h)}{h}=\frac{h}{2}f''(c) \tag{5.80}$$

其中 c 为 x 和 $x-h$ 之间的某个数. 因此, 我们预期向前差分公式与向后差分公式的精度几乎相同.

5.4.1 利用插值的微分

设 $P_n(x)$ 为 $f(x)$ 在 $n+1$ 个节点 $x_0,x_1,\cdots,x_n$ 处的 n 次插值多项式. 为了计算某个点 $x=t$ 处的导数 $f'(x)$, 我们用公式

$$f'(t)\approx P_n'(t) \tag{5.81}$$

通过改变 n 和改变与点 t 有关的节点 $x_0,\cdots,x_n$ 的位置会得到许多不同的公式.

作为公式 (5.81) 的一个特别有用的例子, 取 $n=2$, $t=x_1$, $x_0=x_1-h$, $x_2=x_1+h$, 则

$$\begin{aligned}
P_2(x)&=\frac{(x-x_1)(x-x_2)}{2h^2}f(x_0)+\frac{(x-x_0)(x-x_2)}{-h^2}f(x_1)+\frac{(x-x_0)(x-x_1)}{2h^2}f(x_2)\\
P_2'(x)&=\left(\frac{2x-x_1-x_2}{2h^2}\right)f(x_0)+\left(\frac{2x-x_0-x_2}{-h^2}\right)f(x_1)+\left(\frac{2x-x_0-x_1}{2h^2}\right)f(x_2)\\
P_2'(x_1)&=\left(\frac{x_1-x_2}{2h^2}\right)f(x_0)+\left(\frac{2x_1-x_0-x_2}{-h^2}\right)f(x_1)+\left(\frac{x_1-x_0}{2h^2}\right)f(x_2)\\
&=\frac{f(x_2)-f(x_0)}{2h}
\end{aligned} \tag{5.82}$$

用 x_1-h 和 x_1+h 分别代替 x_0 和 x_2, 由 (5.81) 和 (5.82) 可得

$$f'(x_1)\approx\frac{f(x_1+h)-f(x_1-h)}{2h}\equiv \mathrm{D}_hf(x_1) \tag{5.83}$$

这是 $f(x)$ 的导数的另一种逼近, 称为**中心差分公式**(central difference formula). 下面将证明, (5.83) 比 (5.76) 的 $\mathrm{D}_hf(x)$ 更精确地逼近 $f'(x)$.

一般公式 (5.81) 的误差可以由第 4 章的插值误差公式 (4.53) 得到. 下面的定理给出了主要的结论, 证明过程略去.

定理 5.4.2 假设 $f(x)$ 在区间 $[a,b]$ 上有 $n+2$ 阶连续的导数. 设 $x_0,x_1,\cdots,x_n$ 为 $[a,b]$ 内 $n+1$ 个不同的插值节点, 并设 t 为 $[a,b]$ 内任意给定的数. 则

$$f'(t)-P_n'(t)=\psi_n(t)\frac{f^{(n+2)}(c_1)}{(n+2)!}+\psi_n'(t)\frac{f^{(n+1)}(c_2)}{(n+1)!} \tag{5.84}$$

其中

$$\psi_n(t)=(t-x_0)(t-x_1)\cdots(t-x_n)$$

c_1 和 c_2 是介于 $x_0,x_1,\cdots,x_n,t$ 的最大值和最小值之间的两个未知数.

为了说明这个结果, 我们先推导出 (5.83) 的误差公式. 因为在 (5.83) 的推导过程中 $t=x_1$, 所以 (5.84) 中等式的右端第一项为零. 对 $n=2$, 有

$$\begin{aligned}\psi_2(x)&=(x-x_0)(x-x_1)(x-x_2)\\ \psi_2'(x)&=(x-x_1)(x-x_2)+(x-x_0)(x-x_2)+(x-x_0)(x-x_1)\\ \psi_2'(x_1)&=(x_1-x_0)(x_1-x_2)=-h^2\end{aligned}$$

把它们代入 (5.84), 得到

$$f'(x_1)-\frac{f(x_1+h)-f(x_1-h)}{2h}=-\frac{h^2}{6}f'''(c_2) \tag{5.85}$$

其中 $x_1-h\leqslant c_2\leqslant x_1+h$. 这表明对小的 h 值, 由于公式 (5.83) 的误差随着 h 更迅速地递减, 公式 (5.83) 比公式 (5.76) 更精确.

例 5.4.3 用公式 (5.83) 重新计算表 5-10 中的例题, 其中 $f(x)=\cos(x)$, $x_1=\frac{1}{6}\pi$. 数值结果如表 5-11 所示, 表头为 "D_nf" 的列给出 (5.83). 这些结果证实了 (5.85) 给出的收敛速度, 并且说明公式 (5.83) 通常比前面的逼近公式 (5.76) 更好. ■

表 5-11 用公式 (5.83) 计算 $f(x)=\cos x$ 的数值微分

h	D_hf	误差	比率
0.1	−0.499 167 08	−0.000 832 9	
0.05	−0.499 791 69	−0.000 208 3	4.00
0.025	−0.499 947 92	−0.000 052 08	4.00
0.012 5	−0.499 986 98	−0.000 013 02	4.00
0.006 25	−0.499 996 74	−0.000 003 255	4.00

5.4.2 待定系数法

待定系数法(method of undetermined coefficients) 是一种用来推导数值微分公式、插值公式和数值积分公式的方法. 我们以推导 $f''(x)$ 的逼近公式为例来阐述这种方法.

为了计算在某一点 $x=t$ 处 $f''(x)$ 的近似值, 我们设

$$f''(t) \approx \mathrm{D}_h^{(2)} f(t) \equiv Af(t+h) + Bf(t) + Cf(t-h) \tag{5.86}$$

其中 A, B, C 为待定的常数. 用泰勒多项式近似代替 $f(t-h)$ 和 $f(t+h)$, 有

$$\begin{aligned} f(t-h) &\approx f(t) - hf'(t) + \frac{h^2}{2}f''(t) - \frac{h^3}{6}f'''(t) + \frac{h^4}{24}f^{(4)}(t) \\ f(t+h) &\approx f(t) + hf'(t) + \frac{h^2}{2}f''(t) + \frac{h^3}{6}f'''(t) + \frac{h^4}{24}f^{(4)}(t) \end{aligned} \tag{5.87}$$

包含更多的项会给出 h 的更高次幂; 当 h 很小时, 这些增加项的值应该远小于 (5.87) 中包含的低次项的值. 将其代入 $\mathrm{D}_h^{(2)} f(t)$ 的公式, 按照 h 的乘幂次数合并同类项, 得到

$$\begin{aligned} \mathrm{D}_h^{(2)} f(t) \approx (A+B+C)f(t) + h(A-C)f'(t) + \frac{h^2}{2}(A+C)f''(t) \\ + \frac{h^3}{6}(A-C)f'''(t) + \frac{h^4}{24}(A+C)f^{(4)}(t) \end{aligned} \tag{5.88}$$

要使得对任意的函数 $f(x)$ 都有

$$\mathrm{D}_h^{(2)} f(t) \approx f''(t) \tag{5.89}$$

必须要求

$$\begin{aligned} A+B+C = 0: &\quad f(t)\text{的系数} \\ h(A-C) = 0: &\quad f'(t)\text{的系数} \\ \frac{h^2}{2}(A+C) = 1: &\quad f''(t)\text{的系数} \end{aligned}$$

此方程组的解为

$$A = C = \frac{1}{h^2}, \quad B = -\frac{2}{h^2} \tag{5.90}$$

从而得到

$$\mathrm{D}_h^{(2)} f(t) \equiv \frac{f(t+h) - 2f(t) + f(t-h)}{h^2} \tag{5.91}$$

为了确定 $\mathrm{D}_h^{(2)} f(t)$ 的误差公式, 将 (5.90) 代入到 (5.88), 得到

$$\mathrm{D}_h^{(2)} f(t) \approx f''(t) + \frac{h^2}{12} f^{(4)}(t)$$

在计算这个逼近的过程中没有包含泰勒多项式 (5.87) 中 h 的高次幂项. 因而

$$f''(t) - \frac{f(t+h) - 2f(t) + f(t-h)}{h^2} \approx -\frac{h^2}{12} f^{(4)}(t) \tag{5.92}$$

对小的 h 值, 这是一个十分精确的误差估计. 当然, 在实际问题中我们并不知道 $f^{(4)}(t)$ 的值. 但是由这个误差公式可知, 当 h 减半时, 误差大约减少了 4 倍. 从而可以用它证明, 由理查森外推法可以得到误差和 $f''(t)$ 的更精确的估计值, 参见习题 6.

例 5.4.4 设 $f(x)=\cos(x)$, $t=\frac{1}{6}\pi$, 并且用公式 (5.91) 计算 $f''(t)=-\cos(\frac{1}{6}\pi)$. 计算结果如表 5-12 所示. 注意 "比率" 列的结果和误差公式 (5.92) 是相吻合的. ■

表 5-12 用公式 $D_h^{(2)}f$ 计算 $f(x)=\cos x$ 的数值微分

h	$D_h^{(2)}f$	误差	比率
0.5	−0.848 132 89	−1.789E−2	
0.25	−0.861 524 24	−4.501E−3	3.97
0.125	−0.864 898 35	−1.127E−3	3.99
0.062 5	−0.865 743 53	−2.819E−4	4.00
0.031 25	−0.865 954 93	−7.048E−5	4.00

在推导 (5.91) 的过程中, 假设了近似导数具有 (5.86) 的形式. 我们也可以选择 $f(x)$ 在另外一些点上的值的线性组合, 例如

$$f''(t)\approx Af(t+2h)+Bf(t+h)+Cf(t) \tag{5.93}$$

也可以选取更多的点上的函数值, 例如

$$f''(t)\approx Af(t+3h)+Bf(t+2h)+Cf(t+h)+Df(t) \tag{5.94}$$

更多的自由度能让我们求得 $f''(t)$ 的更加精确的逼近. (5.93) 和 (5.94) 成立的可能性在习题 10 中探究.

许多用待定系数法推导出的公式, 也可以通过对一个适当选取的插值多项式先进行求导再求值来推导. 设想所要求的公式是某些函数值的组合, 然后如 (5.91) 那样导出适当的组合, 这种方法更加简单一些.

5.4.3 函数值误差的影响

对具有解析表达式的函数求导和建立求解微分方程的数值方法来说, 上面给出的公式是很有用的. 然而, 它们对函数值中的误差十分敏感, 尤其是当那些误差与近似求导公式中用到的步长 h 相比不是足够小的时候. 为了研究这一点, 我们要分析逼近 $f''(t)$ 的公式 $D_h^{(2)}f(t)$ 中这种误差的影响.

将 (5.91) 改写为

$$D_h^{(2)}f(x_1)=\frac{f(x_2)-2f(x_1)+f(x_0)}{h^2}\approx f''(x_1) \tag{5.95}$$

其中 $x_2 = x_1 + h$, $x_0 = x_1 - h$. 将计算过程中实际使用的函数值记为 $\hat{f}_0$, $\hat{f}_1$, $\hat{f}_2$, 而

$$f(x_i) - \hat{f}_i = \varepsilon_i, \quad i = 0, 1, 2 \tag{5.96}$$

为函数值的误差. 则实际计算结果为

$$\hat{\mathrm{D}}_h^{(2)} f(x_1) = \frac{\hat{f}_2 - 2\hat{f}_1 + \hat{f}_0}{h^2} \tag{5.97}$$

为研究该结果的误差, 用 $f(x_j) - \varepsilon_j$ 代替 $\hat{f}_j$, $j = 0, 1, 2$, 从而得到

$$\begin{aligned} f''(x_1) - \hat{\mathrm{D}}_h^{(2)} f(x_1) &= f''(x_1) - \frac{[f(x_2) - \epsilon_2] - 2[f(x_1) - \epsilon_1] + [f(x_0) - \epsilon_0]}{h^2} \\ &= \left[f''(x_1) - \frac{f(x_2) - 2f(x_1) + f(x_0)}{h^2}\right] + \frac{\epsilon_2 - 2\epsilon_1 + \epsilon_0}{h^2} \\ &\approx -\frac{h^2}{12} f^{(4)}(x_1) + \frac{\epsilon_2 - 2\epsilon_1 + \epsilon_0}{h^2} \end{aligned} \tag{5.98}$$

上式的最后一步利用了 (5.92).

一般情况下, 误差 $\varepsilon_0, \varepsilon_1, \varepsilon_2$ 是某个区间 $[-\delta, \delta]$ 内的随机数. 若 $\hat{f}_0$, $\hat{f}_1$, $\hat{f}_2$ 的值是实验得到的数据, 那么 δ 是相应的实验误差的一个界. 而如果这些函数值 $\hat{f}_j$ 是通过计算机计算 $f(x)$ 得到的, 则误差 ε_j 就是舍入误差或者截断误差的组合, 并且 δ 是这些误差的一个界. 在上述情况下, 由 (5.98) 可以得到逼近不等式

$$\left|f''(x_1) - \hat{\mathrm{D}}_h^{(2)} f(x_1)\right| \leqslant \frac{h^2}{12}\left|f^{(4)}(x_1)\right| + \frac{4\delta}{h^2} \tag{5.99}$$

因为最后一项是 $4\delta/h^2$, 所以上面的误差界说明：当 $h \to 0$ 时, 误差最终会逐渐增大.

回顾 5.2 节习题 22 的结果, 它与这种误差性态恰好相反. 在习题 22 中, 证明了函数值误差产生的数值积分的变化与积分参数 n(或者等价地, 网格大小 h) 无关. 但是这在数值微分中是不成立的, 正如 (5.98) 或 (5.99) 中指出的那样.

例 5.4.5 用公式 $\hat{\mathrm{D}}_h^{(2)} f(x_1)$ 计算 $f(x) = \cos(x)$ 在 $x_1 = \frac{1}{6}\pi$ 处的导数. 为了显示舍入误差的影响, 把 $f(x_i)$ 舍入到 6 位有效数字作为 $\hat{f}_i$ 的值, 舍入误差满足

$$|\varepsilon_i| \leqslant 5.0 \times 10^{-7} = \delta, \quad i = 0, 1, 2$$

除了这些舍入误差, 公式 $\hat{\mathrm{D}}_h^{(2)} f(x_1)$ 是精确计算的. 表 5-13 给出了数值结果. 在本例中, 误差界 (5.99) 变为:

$$\begin{aligned} \left|f''(x_1) - \hat{\mathrm{D}}_h^{(2)} f(x_1)\right| &\leqslant \frac{h^2}{12}\cos\left(\frac{1}{6}\pi\right) + \left(\frac{4}{h^2}\right)(5 \times 10^{-7}) \\ &\doteq 0.072\,2h^2 + \frac{2 \times 10^{-6}}{h^2} \equiv E(h) \end{aligned} \tag{5.100}$$

当 $h = 0.125$ 时, 误差界 $E(h) \doteq 0.001\,26$, 它与表 5-13 中所给的实际误差相差不多.

表 5-13　$\tilde{D}_h^{(2)} f(x_1)$ 的计算

h	$\tilde{D}_h^{(2)} f(x_1)$	误差
0.5	−0.848 128	−0.017 897
0.25	−0.861 504	−0.004 521
0.125	−0.864 832	−0.001 193
0.062 5	−0.865 536	−0.000 489
0.031 25	−0.865 280	−0.000 745
0.015 625	−0.860 160	−0.005 865
0.007 812 5	−0.851 968	−0.014 057
0.003 906 25	−0.786 432	−0.079 593

误差界 $E(h)$ 表明存在一个最小的 h 值, 记为 $h*$, 使得当 h 小于 $h*$ 时, 误差界便开始增加. 为了求出 $h*$, 令 $E'(h) = 0$, 则 $h*$ 就是这个方程的根. 我们可以算出 $h* \doteq 0.072\,6$, 这与表 5-13 中误差的性态相吻合. 图 5-6 中 $E(h)$ 的图形也证实了这一点. ■

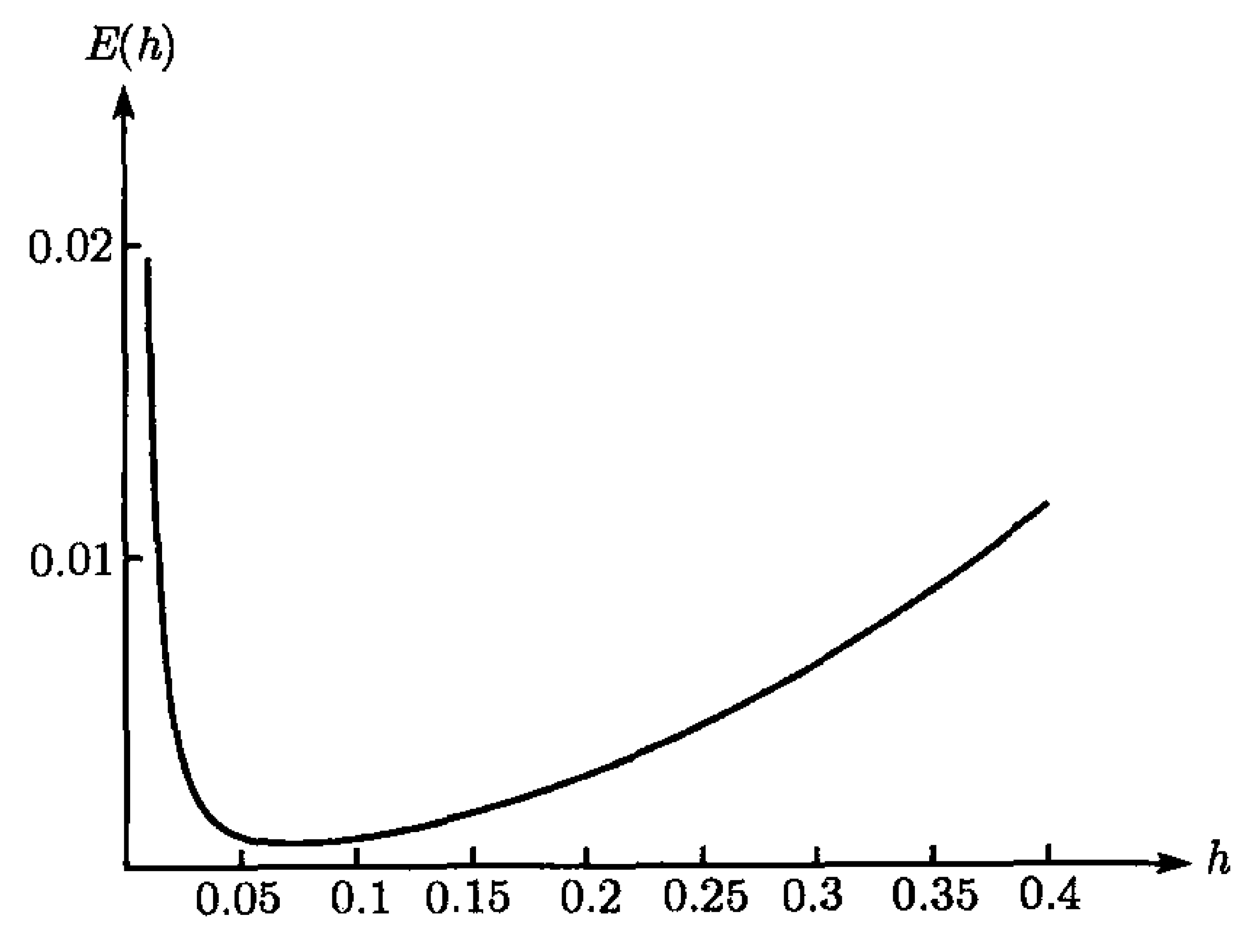


图 5-6　(5.100) 式中 $E(h)$ 的图形

由于函数值误差的敏感性, 我们在使用数值微分公式时必须十分谨慎. 当由实验得到的函数值有较大的误差时, 情况尤其如此; 而这种情况实际上又是十分常见的. 所以对于这种情况, 我们应该使用经过精心准备的计算数值微分的程序包. 这些程序考虑到数据的误差情况, 找到的数值导数在这组数据的误差范围内是精确的. 如果没有这样的程序, 我们应该考虑给出逼近数据的三次样条函数, 再用样条函数的导数作为数据的数值导数. 我们可以用插值方法得到三次样条函数, 而对于误差较大的数据, 最好用数据的**最小二乘逼近**(least squares approximation) 来建立三次样条函数. 最小二乘逼近理论在 7.1 节中给出.

习题

1. 用公式 (5.76) 计算下列函数在给定点处的数值导数 $D_h f(x)$. 取 $h = 0.1, 0.05, 0.025, 0.012\,5, 0.006\,25$. 类似于表 5-10, 计算出误差以及误差减少的比率. 同时, 利用 (5.77) 并用 x 代替 c, 估计误差.
   (a) $f(x) = e^x$ 在 $x = 0$ 处.
   (b) $f(x) = \tan^{-1}(x^2 - x + 1)$ 在 $x = 1$ 处.
   (c) $f(x) = \tan^{-1}(100x^2 - 199x + 100)$ 在 $x = 1$ 处.
2. 用向后差分公式 (5.79) 重做习题 1. 用 x 代替 c, 根据公式 (5.80) 估计误差. 将向后差分的误差与习题 1 中向前差分的误差作比较.
3. 利用 (5.83) 式中的数值导数 $D_h f(x_1)$ 重做习题 1. 利用 (5.85) 并用 x_1 代替 c_2, 估计误差.
4. 设 $h > 0$, $x_j = x_0 + jh$, $j = 0, 1, 2, \cdots, n$. 再设 $P_n(x)$ 为 $f(x)$ 在节点 $x_0, \cdots, x_n$ 处的 n 次插值多项式. 用插值多项式的导数估计 $f'(x_0)$. 当 $n = 1, 2, 3, 4$ 时, 推导实际上含有 $f(x_0), \cdots, f(x_n)$ 的公式, 再给出它们的误差公式.
5. 利用习题 4 定义的 4 次插值多项式 $P_4(x)$ 求 $f'(x_2)$ 的一个逼近, 再求出其误差公式.
6. 误差公式 (5.77), (5.85) 和 (5.92) 几乎都与 h 的幂成正比. 这如 5.2 节中那样证实了理查森外推法的价值. 根据 (5.76) 式, 推导出外推公式

$$f'(x) \approx 2D_h f(x) - D_{2h} f(x)$$

   并证明它的误差比 (5.76) 中的误差收敛到零的速度更快. 基于误差公式 (5.85) 和 (5.92), 分别导出 (5.83) 和 (5.91) 对应的外推公式.
7. 用习题 6 中给出的 (5.76) 的外推公式, 改进表 5-10 中的结果. 根据表 5-10 制作一张外推值的表格, 其中包括外推值的误差和误差减少的比率.
8. 利用 (5.82) 中的多项式 $P_2(x)$, 给出 $f''(x_1)$ 的一个估计, 并分析最终得到的公式.
9. 对习题 1 中的点和函数 $f(x)$, 利用 (5.91) 中的公式 $D_h^{(2)} f(x)$ 估计 $f''(x)$. 再计算它的误差和误差减少的比率. 取 $h = 0.5, 0.25, 0.125, 0.062\,5, 0.031\,25$.
10. (a) 用待定系数法导出公式 (5.93), 要求使误差尽可能小.
    (b) 用待定系数法导出公式 (5.94), 要求使误差尽可能小.
11. 重复表 5-13 中所述的例题, 但取 $f(x) = e^x$ 和 $x_1 = 0$.
12. 仿照 (5.98) 中舍入误差的分析过程, 给出 (5.83) 中 $D_h f(t)$ 的舍入误差的分析.
13. 在单精度算术运算计算机上执行 (5.91) 的数值微分 $D_h^{(2)} f(t)$. 通过逐次减半 h 的值计算 $D_h^{(2)} f(t)$ 及其误差, 继续这个过程, 直到误差开始增加为止.
14. 利用下表中给出的 $f(x)$ 的舍入值, 数值计算 $f''(0.5)$, 步长取 $h = 0.2, 0.1$. 估计由表中元素舍入误差导致的计算结果中误差可能的大小. 试问这是这种情况下误差的主要来源吗?

x	$f(x)$	x	$f(x)$
0.3	7.389 1	0.6	7.614 1
0.4	7.463 3	0.7	7.690 6
0.5	7.538 3		

第 6 章　线性方程组数值求解

线性方程组产生于解决各种不同学科的问题中，这些学科包括数学、统计、物理、生物、社会科学、工程和商业. 它们在解决现实世界问题中直接产生，并且也在解决其他问题 (例如求解非线性方程组) 的过程中出现. 微分方程边值问题和初边值问题的数值解产生了大量的线性方程组，特别是大规模线性方程组. 在这一章中我们将考察求解线性方程组的一些经典方法，包括如高斯消元法那样的直接方法，以及雅可比和高斯–赛德尔那样的迭代法.

我们在第一节给出与线性方程组有关的记号及理论知识，然后在 6.3 节讨论求解线性方程组的一般方法. 利用矩阵代数的工具能使线性方程组的表述与讨论较为简单，有关矩阵代数的内容在 6.2 节介绍. 在 6.4 节与 6.5 节中我们利用矩阵代数扩展 6.3 节中解线性方程组的思想，并且考虑特殊类型的线性方程组及其舍入误差的影响. 6.6 节介绍解线性方程组的迭代法，给出一些较常见的简单的迭代过程.

6.1　线性方程组

在初等代数中研究的一个问题是解形如

$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases} \tag{6.1}$$

的线性方程组，其中系数 $a, b, \cdots, f$ 是已知常数，任务是找出未知值 x, y. 在这一章中，我们考查包含更多方程和未知数的线性方程组求解问题.

为了写出我们将研究的最一般的线性方程组，我们必须用更方便的记号来代替方程组 (6.1) 的表述方式. 设 n 是正整数，则含 n 个未知量 $x_1, x_2, \cdots, x_n$ 的 n 个线性方程的方程组的一般形式是

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{6.2}$$

我们用符号 a_{ij} 给出第 i 个方程的相应于第 j 个未知量的系数. 在有些场合，为了避免可能的混淆，也使用记号 $a_{i,j}$. 右端的 $b_1, \cdots, b_n$ 是已知数，我们要计算未知量 $x_1, x_2, \cdots, x_n$. 称这样的线性方程组为 n 阶的.

例 6.1.1 对于 $1 \leqslant i,j \leqslant n$, 用

$$a_{ij} = \max\{i,j\} \tag{6.3}$$

定义 (6.2) 的系数, 再用

$$b_1 = b_2 = \cdots = b_n = 1 \tag{6.4}$$

定义右端. 相应的线性方程组 (6.2) 的解是

$$x_1 = x_2 = \cdots = x_{n-1} = 0, \quad x_n = \frac{1}{n} \tag{6.5}$$

可以证明这是线性方程组的唯一解. 作为具体的例子, 设 $n = 3$. 则线性方程组是

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 1 \\ 2x_1 + 2x_2 + 3x_3 = 1 \\ 3x_1 + 3x_2 + 3x_3 = 1 \end{cases} \tag{6.6}$$

它的解是

$$x_1 = x_2 = 0, \quad x_3 = \frac{1}{3}.$$

■

对于阶数较小的线性方程组, 如 (6.1) 和 (6.6), 有可能用初等代数中学过的方法通过手算或借助于计算器求解它们. 然而, 在大多数应用中产生的线性方程组通常阶数较高, 少则几十, 多到几百万. 显然, 不可能用人工方法求解这么大的线性方程组. 为求它们的解我们需要利用数值方法. 为此, 最方便的是用矩阵/向量记号表示线性方程组, 并且对它们的数值处理使用相应的矩阵/向量算术运算.

线性方程组 (6.2) 由系数 a_{ij} 和右端的常数 b_i 完全确定. 我们把这些系数看作一个**矩阵**的元素:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{6.7}$$

称 a_{ij} 是矩阵 $\boldsymbol{A}$ 的 (i,j) 元素. 类似地, 把右端的常数 b_i 看作向量

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{6.8}$$

的元素.

字母 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 是矩阵和向量的名字. a_{ij} 的下标表明这是矩阵 $\boldsymbol{A}$ 的第 i 行第 j 列的元素. 类似地, 把解 $x_1, x_2, \cdots, x_n$ 写成

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{6.9}$$

例 6.1.2　对于线性方程组 (6.6),

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 3 & 3 & 3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{3} \end{bmatrix} \tag{6.10}$$

在 MATLAB 中, 矩阵 $\boldsymbol{A}$ 可以用下列任何方式产生:

```
A = [1 2 3; 2 2 3; 3 3 3];
A = [1, 2, 3; 2, 2, 3; 3, 3, 3];
A = [1 2 3
     2 2 3
     3 3 3];
```

并且在最后一种情况可以用逗号来代替每一行元素之间的空格. 向量 $\boldsymbol{b}$ 可以用下列方式产生

```
b = ones(3,1);
```

更一般地, 对于由 (6.3) 和 (6.4) 定义的线性方程组 (6.2), 我们产生矩阵如下:

```
A = zeros(n,n);
for i = 1:n
   A(i,1:i) = i;
   A(i,i+1:n) = i+1:n;
end
```

并且用

```
b = ones(n,1);
```

产生向量 $\boldsymbol{b}$. ■

用 (6.7)~(6.9) 中引进的记号, 线性方程组 (6.2) 可写成紧凑形式

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{6.11}$$

具备一些线性代数知识的读者会马上认识到 (6.11) 左端是矩阵 $\boldsymbol{A}$ 乘以向量 $\boldsymbol{x}$, 并且 (6.11) 表示两个向量 $\boldsymbol{Ax}$ 和 $\boldsymbol{b}$ 之间的等式. 方程组 (6.11) 由 n 个关系式组成, 并且对于 $i=1,\cdots,n$, 第 i 个关系式就是方程组 (6.2) 中的第 i 个方程.

由于本书不假定读者掌握线性代数方面的知识, 所以下一节将复习矩阵和向量的算术运算及其性质.

习题

1. **(a)** 考虑三次多项式插值问题

$$p(x_i)=y_i, \quad i=0,1,2,3$$

其中多项式 p 的次数 $\deg(p)\leqslant 3$, 并且 x_0, x_1, x_2, x_3 互不相同. 把寻找 $p(x)$ 问题转化为一个相应的线性方程组求解问题.

提示: 记

$$p(x)=a_0+a_1x+a_2x^2+a_3x^3$$

并确定 a_0, a_1, a_2, a_3. 利用插值条件得到包含 $a_0,\cdots,a_3$ 的方程.

(b) 将从 (a) 得到的方程组表示成 (6.11) 的形式, 确定矩阵 $\boldsymbol{A}$ 以及向量 $\boldsymbol{b}$ 和 $\boldsymbol{x}$.

2. 重复习题 1, 但用下列埃尔米特插值条件:

$$p(x_i)=y_i, \quad p'(x_i)=y_i', \quad i=1,2$$

这里 $\{x_i,y_i,y_i'\}$ 已知, 且 $x_1\neq x_2$.

3. 回忆插值三次样条函数的方程 (4.64) 与 (4.65). 假设插值结点是均匀间隔的, 间隔为 h, 对相应的线性方程组给出系数矩阵和右端向量.

4. 考虑矩阵

$$\boldsymbol{A}=\begin{bmatrix} 3 & 1 & 0 & \cdots & 0 \\ 1 & 3 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & 3 & 1 \\ 0 & \cdots & 0 & 1 & 3 \end{bmatrix}$$

和向量

$$\boldsymbol{b}=\begin{bmatrix} 4 \\ 5 \\ 5 \\ \vdots \\ 5 \\ 4 \end{bmatrix}, \quad \boldsymbol{x}=\begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

给出与 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 相对应的 n 阶线性方程组 (6.2). 证明给定的 $\boldsymbol{x}$ 是这线性方程组的解.

5. 用 MATLAB 定义下列矩阵 $\boldsymbol{A}$ 和向量 $\boldsymbol{b}$:

$$a_{ij}=\begin{cases}\dfrac{i}{j},1\leqslant i\leqslant j\leqslant n\\ \dfrac{j}{i},1\leqslant j\leqslant i\leqslant n\end{cases},\quad b_i=\begin{cases}-1, & i\text{ 是偶数}\\ 1, & i\text{ 是奇数}\end{cases}$$

6. 对于下列矩阵 $\boldsymbol{A}$ 和向量 $\boldsymbol{b}$ 重复习题 5:

$$a_{1j}=a_{j1}=\beta,\quad j=1,2,\cdots,n$$

$$a_{ij}=a_{i-1,j}+a_{i,j-1},\quad i,j=2,3,\cdots,n$$

$$b_i=(-1)^{i-1}/i,\quad i=1,\cdots,n$$

设 β 是程序的输入参数, $\beta\neq 0$. 给出 $\beta=1$ 时的矩阵.

7. 用 MATLAB 建立线性方程组 (4.113) 的系数矩阵 H_{n+1}. 注意

$$(H_{n+1})_{i,j}=\frac{1}{i+j-1},\quad 1\leqslant i,j\leqslant n+1$$

6.2 矩阵算术运算

本节我们正式定义矩阵和向量, 以及它们的算术运算和相关性质, 这对线性方程组数值解是有用的.

矩阵是一个长方形数组

$$\boldsymbol{B}=\begin{bmatrix} b_{11} & \cdots & b_{1n}\\ \vdots & \ddots & \vdots\\ b_{m1} & \cdots & b_{mn}\end{bmatrix} \tag{6.12}$$

如果它有 m 行 n 列, 则称它为 $m\times n$ 阶矩阵. n 行 n 列方阵称为 n 阶矩阵. 单行或单列组成的矩阵分别称为**行矩阵**或**列矩阵**. 它们常常被称为行向量或列向量, 或被简称为**向量**. 这是因为它们可以用从空间原点到由行矩阵或列矩阵给出的坐标点的向量来刻画. 这样, 可以从几何的观点来研究矩阵. 不过, 我们不作这样的讨论. 由于向量是特殊类型的长方形矩阵, 因此一般长方形矩阵的运算和性质对向量也适用.

作为记号, 我们用大写字母表示矩阵; 而带有两个下标的相应小写字母通常表示该矩阵的元素. 用小写字母表示向量, 并且用带有一个下标的同样小写字母表示该向量的元素; (6.7) 到 (6.9) 式给出了这样的例子. 如果两个矩阵阶数相同, 并且对应元素相等, 则称这两个矩阵为相等的. 一般矩阵 (6.12) 的转置为

$$\boldsymbol{B}^{\mathrm{T}}=\begin{bmatrix} b_{11} & \cdots & b_{m1}\\ \vdots & \ddots & \vdots\\ b_{1n} & \cdots & b_{mn}\end{bmatrix} \tag{6.13}$$

它通过交换 $\boldsymbol{B}$ 的行和列得到; $\boldsymbol{B}^{\mathrm{T}}$ 的阶数为 $n \times m$. 特别地, 行向量的转置为列向量, 反之亦然. 在 MATLAB 中, 一旦定义矩阵 $\boldsymbol{A}$, 则它的转置通过 $\boldsymbol{A}'$ 得到.

例 6.2.1

$$\boldsymbol{A} \equiv \begin{bmatrix} a & b & 1 \\ 2 & c & d \end{bmatrix} = \begin{bmatrix} 5 & 6 & 1 \\ 2 & 2 & a+1 \end{bmatrix}$$

可推出

$$a = 5, \quad b = 6, \quad c = 2, \quad d = a + 1 = 6$$

并且

$$\boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} a & 2 \\ b & c \\ 1 & d \end{bmatrix}$$

$\boldsymbol{A}$ 为 2×3 阶, 而 $\boldsymbol{A}^{\mathrm{T}}$ 为 3×2 阶. ■

6.2.1 算术运算

有三种与矩阵有关的基本算术运算. 第一种算术运算是数乘矩阵. 设 $\boldsymbol{B}$ 是 (6.12) 那样的 $m \times n$ 阶矩阵, 并且设 a 是一个任意的实数. 则 $a\boldsymbol{B}$ 是下式定义的 $m \times n$ 阶矩阵

$$a\boldsymbol{B} = \begin{bmatrix} ab_{11} & \cdots & ab_{1n} \\ \vdots & \ddots & \vdots \\ ab_{m1} & \cdots & ab_{mn} \end{bmatrix} \tag{6.14}$$

$a\boldsymbol{B}$ 中的元素由 a 乘以 $\boldsymbol{B}$ 的相应元素得到. 特别地, 取 $a = -1$,

$$(-1)\boldsymbol{B} = \begin{bmatrix} -b_{11} & \cdots & -b_{1n} \\ \vdots & \ddots & \vdots \\ -b_{m1} & \cdots & -b_{mn} \end{bmatrix}$$

这个矩阵称为 $\boldsymbol{B}$ 的负矩阵并用 $-\boldsymbol{B}$ 表示, 即

$$-\boldsymbol{B} = (-1)\boldsymbol{B}$$

第二种算术运算是同阶矩阵相加. 设 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是 $m \times n$ 阶矩阵. 则 $\boldsymbol{A} + \boldsymbol{B}$ 是如下 $m \times n$ 阶矩阵

$$\boldsymbol{A} + \boldsymbol{B} = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

$\boldsymbol{A}+\boldsymbol{B}$ 的 (i,j) 元素是 $a_{ij}+b_{ij}$. 对矩阵加法有零矩阵. 定义所有元素都为零的 $m\times n$ 阶矩阵为 $m\times n$ 阶零矩阵, 用 $\boldsymbol{O}_{m\times n}$ 表示, 或更经常地, 记为 $\boldsymbol{O}$. 对任何矩阵 $\boldsymbol{A}$ 都有性质

$$\boldsymbol{A}+\boldsymbol{O}=\boldsymbol{O}+\boldsymbol{A}=\boldsymbol{A}$$

零矩阵的阶数由上下文确定.

利用矩阵的求和与标量乘法, 我们可以定义同阶矩阵的减法:

$$\boldsymbol{A}-\boldsymbol{B}=\boldsymbol{A}+(-\boldsymbol{B})$$

对任何矩阵 $\boldsymbol{A}$, 成立

$$\boldsymbol{A}-\boldsymbol{A}=\boldsymbol{A}+(-\boldsymbol{A})=\boldsymbol{O}$$

例 6.2.2

$$-5\begin{bmatrix}2&3\\6&-1\end{bmatrix}=\begin{bmatrix}-10&-15\\-30&5\end{bmatrix}$$

$$\begin{bmatrix}3&2&1\\1&2&3\end{bmatrix}+\begin{bmatrix}1&2&3\\3&2&1\end{bmatrix}=\begin{bmatrix}4&4&4\\4&4&4\end{bmatrix}=4\begin{bmatrix}1&1&1\\1&1&1\end{bmatrix}$$

$$\begin{bmatrix}3&2&1\\1&2&3\end{bmatrix}-\begin{bmatrix}1&2&3\\3&2&1\end{bmatrix}=\begin{bmatrix}2&0&-2\\-2&0&2\end{bmatrix}=2\begin{bmatrix}1&0&-1\\-1&0&1\end{bmatrix}$$

$$\begin{bmatrix}2&3\\-1&5\end{bmatrix}+\begin{bmatrix}-2&-3\\1&-5\end{bmatrix}=\begin{bmatrix}0&0\\0&0\end{bmatrix}=0$$ ■

第三种算术运算是较复杂的矩阵乘法. 设 $\boldsymbol{A}$ 为 $m\times n$ 阶矩阵且 $\boldsymbol{B}$ 为 $n\times p$ 阶矩阵. 则 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的乘积 $\boldsymbol{C}=\boldsymbol{AB}$ 是一个 $m\times p$ 阶矩阵, 它的一般元素 c_{ij} 定义为

$$c_{ij}=\sum_{k=1}^{n}a_{ik}b_{kj},\quad 1\leqslant i\leqslant m,\qquad 1\leqslant j\leqslant p \tag{6.15}$$

这是 $\boldsymbol{A}$ 的第 i 行与 $\boldsymbol{B}$ 的第 j 列对应元素的乘积之和. 如果 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是方阵, 那么 $\boldsymbol{AB}$ 是同阶方阵. $m\times n$ 阶矩阵 $\boldsymbol{A}$ 与 n 维列向量 $\boldsymbol{b}$ 的乘积是 m 维列向量 $\boldsymbol{Ab}$.

矩阵乘法定义把线性函数的复合联系起来. 把 n 维向量映射成 m 维向量的线性函数

$$f(\boldsymbol{x})=\begin{bmatrix}a_{11}x_1+\cdots+a_{1n}x_n\\ \vdots\\ a_{m1}x_1+\cdots+a_{mn}x_n\end{bmatrix},\quad 其中\quad \boldsymbol{x}=\begin{bmatrix}x_1\\ \vdots\\ x_n\end{bmatrix}$$

可以表示为矩阵和向量的乘积

$$f(\boldsymbol{x})=\boldsymbol{Ax}$$

其中

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

设线性函数 f 和 g 由下面定义

$$f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}, \quad \text{其中} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$g(\boldsymbol{y}) = \boldsymbol{B}\boldsymbol{y}, \quad \text{其中} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix}$$

则与复合函数 $h(\boldsymbol{y}) \equiv f(g(\boldsymbol{y}))$ 联系起来的矩阵是 (6.15) 中定义的 $\boldsymbol{C} = \boldsymbol{AB}$.

例 6.2.3 (a)

$$\begin{bmatrix} 2 & 3 & 2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & -1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 14 & -1 \\ 0 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 2 & -1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 2 \\ -1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 2 \\ 5 & 4 & 5 \\ 5 & 11 & 5 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix}$$

(c)

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 3 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3x_1 + 2x_2 + x_3 \\ x_1 + 2x_2 + 3x_3 \\ x_1 - x_2 + x_3 \end{bmatrix}$$

■

正如最后一个例子所显示的, 线性方程组可以利用矩阵乘法写成矩阵方程. 例如, 方程组

$$\begin{cases} x_1 + 2x_2 + x_3 = 0 \\ 2x_1 + 2x_2 + 3x_3 = 3 \\ -x_1 - 3x_2 = 2 \end{cases}$$

可以写成

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ -1 & -3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 2 \end{bmatrix}$$

读者可对左边进行矩阵乘法, 观察两边对应元素相等得到的方程组.

根据前节末尾的注释, 一般的线性方程组

$$\begin{cases} a_{11}x_{1+}\cdots + a_{1n}x_n = b_1 \\ \qquad\qquad \vdots \\ a_{n1}x_{1+}\cdots + a_{nn}x_n = b_n \end{cases} \tag{6.16}$$

可以简洁地写成

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{6.17}$$

当用这种方式书写时, 许多结果概念上变得较清楚. 同样, 可将矩阵理论和线性代数中的结果应用于 (6.17), 既得到关于求解线性方程组 (6.16) 的理论结果又得到计算方法.

6.2.2 初等行运算

在下一节讨论的求解线性方程组的高斯消元法, 最方便的是用矩阵形式来描述. 作为准备, 我们对一般长方形矩阵介绍以下三种初等行运算.

(i) 交换两行.

(ii) 用一个非零标量乘某行.

(iii) 某行非零倍数加到另一行.

例 6.2.4 考虑长方形矩阵

$$\boldsymbol{A} = \begin{bmatrix} 3 & 3 & 3 & 1 \\ 2 & 2 & 3 & 1 \\ 1 & 2 & 3 & 1 \end{bmatrix}$$

将第 2 行的 (–1) 倍加到第 1 行, 再将第 3 行的 (–1) 倍加到第 2 行, 得到矩阵:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 1 \end{bmatrix}$$

第 2 行的 (–1) 倍加到第 1 行, 同样也加到第 3 行:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 3 & 1 \end{bmatrix}$$

第 1 行的 (−2) 倍加到第 3 行：

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 \end{bmatrix}$$

交换第 1 行和第 2 行：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 1 \end{bmatrix}$$

最后, 我们用 $\frac{1}{3}$ 乘第 3 行：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{3} \end{bmatrix}$$

这是从原始矩阵 $\boldsymbol{A}$ 通过初等行运算后得到的一个矩阵. ■

6.2.3 矩阵的逆

对所有实数 x, 数字 1 满足

$$1 \cdot x = x \cdot 1 = x$$

对于矩阵乘法, 与数字 1 相应的是**恒等矩阵** (identity matrix) $\boldsymbol{I}$, 在有些参考书中也称为**单位矩阵** (unit matrix). 设 $n \geqslant 1$, 定义 $\boldsymbol{I}_n$ 是 n 阶方阵, 它的 (i,j) 元素为

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

对于 $n = 3$,

$$\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

恒等矩阵的最重要性质是

$$\boldsymbol{A}\boldsymbol{I}_n = \boldsymbol{A}, \quad \boldsymbol{I}_m\boldsymbol{A} = \boldsymbol{A} \tag{6.18}$$

其中 $\boldsymbol{A}$ 是任意 $m \times n$ 阶矩阵. 这是类似于上面关于实数 1 的恒等式.

对每个正整数 n, 有一个阶数为 n 的恒等矩阵 $\boldsymbol{I}_n$. 我们通常省略下标 n, 而用 $\boldsymbol{I}$ 表示任何可能阶的恒等矩阵, 阶数的选择由上下文决定. 例如, 可将 (6.18) 写成

$$\boldsymbol{AI} = \boldsymbol{IA} = \boldsymbol{A}$$

我们还可以将数的倒数的概念推广到一些方阵. 设 $\boldsymbol{A}$ 是一个方阵. 如果存在一个方阵 $\boldsymbol{B}$ 使得

$$\boldsymbol{AB} = \boldsymbol{I}, \quad \boldsymbol{BA} = \boldsymbol{I} \tag{6.19}$$

则称 $\boldsymbol{B}$ 为 $\boldsymbol{A}$ 的**逆** (inverse). 后面将说明如果 $\boldsymbol{A}$ 有逆, 则它只有一个逆. 类似于一个实数的倒数记号, $\boldsymbol{A}$ 的逆用 $\boldsymbol{A}^{-1}$ 表示. 为验证 $\boldsymbol{B}$ 是 $\boldsymbol{A}$ 的逆, 只需要证明 (6.19) 中的任何一个等式, 因为 (6.19) 中每一个式子都可以从另一个导出.

例 6.2.5 (a)

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \quad \boldsymbol{A}^{-1} = \begin{bmatrix} \frac{3}{4} & -\frac{1}{2} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{3}{4} \end{bmatrix}$$

(b) 设

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

则可以证明 $\boldsymbol{A}$ 有逆当且仅当 $ad - bc \neq 0$, 此时

$$\boldsymbol{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{6.20}$$

在 (a) 和 (b) 中, 为检验关于 $\boldsymbol{A}^{-1}$ 的公式, 只要将相应的两个矩阵相乘看看是否得到恒等矩阵. ■

对于实数, 数 b 有一个倒数 (或逆) 当且仅当 $b \neq 0$. 对于矩阵, 这个情况的推广如下.

定理 6.2.6 设 $\boldsymbol{A}$ 是一个方阵. 则 $\boldsymbol{A}$ 有逆当且仅当 $\det(\boldsymbol{A}) \neq 0$.

在这个定理中, $\det(\boldsymbol{A})$ 表示 $\boldsymbol{A}$ 的**行列式** (determinant). 许多人在初等代数中关于求解方程组 (6.16) 时已学过行列式, 因此我们省略它的任何理论讨论. 满足 $\det(\boldsymbol{A}) = 0$ 的矩阵 $\boldsymbol{A}$ 称为**奇异矩阵**, 而满足 $\det(\boldsymbol{A}) \neq 0$ 的矩阵 $\boldsymbol{A}$ 称为**非奇异矩阵**.

对于阶数 $n = 2$ 的矩阵

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

它的行列式为

$$\det(\boldsymbol{A}) = a_{11}a_{22} - a_{12}a_{21} \tag{6.21}$$

6.2.4 矩阵代数法则

实数具有的许多算术运算法则对矩阵仍然适用, 但也有一些重要的差别. 我们列出这些法则, 并对其中一些给出证明或说明. 在这些法则中, 除了涉及逆阵和行列式时以外, 矩阵不必是方阵. 假设在一个关系式中矩阵的阶数是相容的, 使得列出的算术运算表达式有意义. 例如, 在 (6.24) 中, $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是同阶长方矩阵.

$$(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C} = \boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C}) \tag{6.22}$$

$$(\boldsymbol{AB})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{BC}) \tag{6.23}$$

$$\boldsymbol{A}+\boldsymbol{B} = \boldsymbol{B}+\boldsymbol{A} \tag{6.24}$$

$$\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{AB}+\boldsymbol{AC} \tag{6.25}$$

$$(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{C} = \boldsymbol{AC}+\boldsymbol{BC} \tag{6.26}$$

$$(\boldsymbol{AB})^{\mathrm{T}} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} \tag{6.27}$$

$$(\boldsymbol{A}+\boldsymbol{B})^{\mathrm{T}} = \boldsymbol{A}^{\mathrm{T}}+\boldsymbol{B}^{\mathrm{T}} \tag{6.28}$$

$$(c\boldsymbol{A})^{-1} = \frac{1}{c}\boldsymbol{A}^{-1}, \quad c = \text{ 非零常数} \tag{6.29}$$

$$(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \tag{6.30}$$

$$\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B}) \tag{6.31}$$

$$\det(\boldsymbol{A}^{\mathrm{T}}) = \det(\boldsymbol{A}) \tag{6.32}$$

$$\det(c\boldsymbol{A}) = c^n\det(\boldsymbol{A}), \quad A \text{ 的阶 } = n, \quad c = \text{ 常数} \tag{6.33}$$

性质 (6.22) 和 (6.23) 是结合律, (6.24) 是交换律, (6.25) 和 (6.26) 是分配律. 在 (6.29) 和 (6.30) 中隐含假设 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是同阶可逆方阵. (6.29) 表明对任何常数 $c \neq 0$, $c\boldsymbol{A}$ 也可逆且它的逆是 $\frac{1}{c}\boldsymbol{A}^{-1}$. 类似地, (6.30) 表明 $\boldsymbol{AB}$ 可逆, 其逆为 $\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$. 性质 (6.31)、(6.32) 以及 (6.33) 给出矩阵行列式的结论.

作为一个例子, 我们来证明 (6.23). 设 $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ 的阶数分别为 $m \times n$, $n \times p$, $p \times q$. 则 $\boldsymbol{AB}$ 为 $m \times p$ 阶的, 而 $\boldsymbol{BC}$ 为 $n \times q$ 阶的. 下面 $\boldsymbol{D}_{ij}$ 表示矩阵 $\boldsymbol{D}$ 的 (i,j) 元素, 于是

$$[(\boldsymbol{AB})\boldsymbol{C}]_{ij} = \sum_{l=1}^{p}(\boldsymbol{AB})_{il}\boldsymbol{C}_{lj} = \sum_{l=1}^{p}\left[\sum_{k=1}^{n}\boldsymbol{A}_{ik}\boldsymbol{B}_{kl}\right]\boldsymbol{C}_{lj}$$

$$[\boldsymbol{A}(\boldsymbol{BC})]_{ij} = \sum_{k=1}^{n} \boldsymbol{A}_{ik}(\boldsymbol{BC})_{kj} = \sum_{k=1}^{n} \boldsymbol{A}_{ik}\left[\sum_{l=1}^{p} \boldsymbol{B}_{kl}\boldsymbol{C}_{lj}\right]$$

对于 $1 \leqslant k \leqslant n, 1 \leqslant l \leqslant p$, 将所有乘积 $\boldsymbol{A}_{ik}\boldsymbol{B}_{kl}\boldsymbol{C}_{lj}$ 之和重排, 得到这两个式子的右边是相同的. 这就证明了 (6.23).

例 6.2.7 如果 $\boldsymbol{A}$ 有逆阵, 则逆矩是唯一的. 为了证明这个结论, 假设存在两个逆阵 $\boldsymbol{B}_1$ 和 $\boldsymbol{B}_2$ 具有性质

$$\boldsymbol{B}_1\boldsymbol{A} = \boldsymbol{A}\boldsymbol{B}_1 = \boldsymbol{I}, \quad \boldsymbol{B}_2\boldsymbol{A} = \boldsymbol{A}\boldsymbol{B}_2 = \boldsymbol{I}$$

让我们考察乘积 $\boldsymbol{B}_1\boldsymbol{A}\boldsymbol{B}_2$. 首先, 利用结合律

$$(\boldsymbol{B}_1\boldsymbol{A})\boldsymbol{B}_2 = \boldsymbol{B}_1(\boldsymbol{A}\boldsymbol{B}_2)$$

再由 $\boldsymbol{B}_1$ 和 $\boldsymbol{B}_2$ 的假设得,

$$(\boldsymbol{B}_1\boldsymbol{A})\boldsymbol{B}_2 = \boldsymbol{I}\boldsymbol{B}_2 = \boldsymbol{B}_2$$

$$\boldsymbol{B}_1(\boldsymbol{A}\boldsymbol{B}_2) = \boldsymbol{B}_1\boldsymbol{I} = \boldsymbol{B}_1$$

把这些式子合在一起证明 $\boldsymbol{B}_1 = \boldsymbol{B}_2$. 因此, $\boldsymbol{A}$ 只有一个逆阵. ■

例 6.2.8 设

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

则

$$\det(\boldsymbol{A}) = -2, \quad \det(\boldsymbol{B}) = 2$$

$$\boldsymbol{AB} = \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix}$$

$$\det(\boldsymbol{AB}) = -4 = \det(\boldsymbol{A})\det(\boldsymbol{B})$$

这样, 对给定的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$, 我们说明了 (6.31) 式. ■

从前面列出的法则中找不到乘法的交换律:

$$\boldsymbol{AB} = \boldsymbol{BA} \tag{6.34}$$

因为这个等式对大多数方阵 $\boldsymbol{A}$ 和方阵 $\boldsymbol{B}$ 不成立. 满足 (6.34) 式的两个矩阵称为相互可交换.

例 6.2.9 对上一个例子中的矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$, 有

$$\boldsymbol{AB} = \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix}, \quad \boldsymbol{BA} = \begin{bmatrix} -2 & -2 \\ 4 & 6 \end{bmatrix}$$

矩阵 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$ 不满足交换律. ■

6.2.5 线性方程组的可解性理论

考虑线性方程组 (6.17), $\boldsymbol{Ax}=\boldsymbol{b}$, 这里 $\boldsymbol{A}$ 是 n 阶方阵. 在线性方程组 (6.17) 的可解性理论中, 令所有常量 $b_i=0$ 得到的特殊线性方程组扮演一个重要的角色. 这样的线性方程组称为齐次的, 所有其他线性方程组称为非齐次的. 下列定理总结有关线性方程组的一些重要的理论结果.

定理 6.2.10 设 n 是正整数, 而 $\boldsymbol{A}$ 是 n 阶方阵. 则下面关于线性方程组 (6.17) 的陈述是等价的.

S1. 对每个右边 $\boldsymbol{b}$, 方程组 (6.17) 恰好有一个解 $\boldsymbol{x}$.

S2. 对每个右边 $\boldsymbol{b}$, 方程组 (6.17) 至少有一个解 $\boldsymbol{x}$.

S3. (6.17) 的齐次方程组恰好有一个解

$$x_1=x_2=\cdots=x_n=0$$

S4. $\det(\boldsymbol{A})\neq 0$.

本章考虑非奇异线性方程组的解. 定理中的 (S1) 是说, 对任意的右边常量 $b_1,\cdots,b_n$, 方程组都有唯一解 $x_1,\cdots,x_n$. 定理的余下部分 (S2) 到 (S4) 等价于 (S1); 为了对给定的方程组证明 (S1) 成立, 通常证明 (S2)、(S3) 或 (S4) 更容易. 例如, 涉及插值多项式存在性和唯一性的问题常常先把它们变为 (6.17) 形式的线性方程组问题来处理; 然后, 对这些方程组证明 (S3) 成立. 这里我们省略这样的例子, 但它们通常出现在高等数值分析中.

利用 (S4) 可知, 2 阶线性方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2=b_1\\ a_{21}x_1+a_{22}x_2=b_2\end{cases}$$

对所有右端 b 有唯一解当且仅当

$$a_{11}a_{22}-a_{12}a_{21}\neq 0$$

这是一个容易检验的条件. 对阶数较大的线性方程组, 在决定与 $\boldsymbol{A}$ 相关的线性方程组的可解性时有时也使用 (S4).

回顾前面的定理 6.2.6, 我们看到 $\boldsymbol{A}^{-1}$ 存在等价于定理 6.2.10 的陈述 (S1) 到 (S4) 中的任一个成立. 因此, $\boldsymbol{A}$ 可逆当且仅当线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 对任何右端列向量 $\boldsymbol{b}$ 有唯一解.

习题

1. 进行矩阵运算, 将结果写成单个矩阵:

(a)
$$2\begin{bmatrix} 1 & 0 \\ -1 & 3 \end{bmatrix}+\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 2 & 3 \\ 1 & 3 \end{bmatrix}$$

(b)
$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} a & b \\ 0 & c \end{bmatrix}$$

(c)
$$\boldsymbol{A}=\boldsymbol{I}_3-2\omega\omega^{\mathrm{T}},\quad \omega^{\mathrm{T}}=\left[\tfrac{1}{3},\tfrac{2}{3},\tfrac{2}{3}\right],\quad \boldsymbol{B}=\boldsymbol{A}^2$$

(d)
$$\boldsymbol{A}=\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix},\quad \boldsymbol{B}=\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$$

(e)
$$\boldsymbol{A}=\begin{bmatrix} 2 & 1 & 0 & \cdots & 0 \\ 1 & 2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & 2 & 1 \\ 0 & \cdots & 0 & 1 & 2 \end{bmatrix},\quad \boldsymbol{B}=\boldsymbol{A}^2$$

2. 设 $\boldsymbol{A}$ 是任意的 3 阶方阵, 并设

$$\boldsymbol{D}=\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

计算 $\boldsymbol{AD}$ 和 $\boldsymbol{DA}$. 矩阵 $\boldsymbol{D}$ 称为**对角矩阵**. 给出一般矩阵乘以对角矩阵的一个简单法则.

3. 设 $\boldsymbol{A}$ 是 n 阶方阵, 满足

$$a_{ij}=0,\quad i>j$$

这种矩阵称为**上三角矩阵**. 证明这种矩阵的和与积也是上三角矩阵. 如果逆存在, 它的逆是上三角矩阵吗?

4. 如果方阵 $\boldsymbol{A}$ 的转置 $\boldsymbol{A}^{\mathrm{T}}$ 是上三角矩阵, 则称方阵 $\boldsymbol{A}$ 为**下三角矩阵**. 利用这个定义、前面习题的结果以及矩阵转置的性质, 证明下三角矩阵的和与积也是下三角矩阵.

5. 设 $\boldsymbol{w}$ 是满足 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}=1$ 的列向量. 乘积 $\boldsymbol{A}=\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}$ 是方阵. 证明 $\boldsymbol{A}^2=\boldsymbol{A}$.

6. 如果 $\boldsymbol{B}^{\mathrm{T}}=\boldsymbol{B}$, 则称矩阵 B 为**对称矩阵**. 写出所有 2×2 阶对称矩阵的一般形式.

7. 把习题 5 和习题 6 结合起来, 定义一个新矩阵

$$\boldsymbol{B}=\boldsymbol{I}-2\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}$$

这里 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}=1$. 证明 $\boldsymbol{B}$ 是对称矩阵并且 $\boldsymbol{B}^2=\boldsymbol{I}$. $\boldsymbol{B}^{-1}$ 是什么样的矩阵?

8. 设 $\boldsymbol{A}$ 为 $m\times n$ 阶, $\boldsymbol{B}$ 为 $n\times p$ 阶. 求计算 $\boldsymbol{AB}$ 的运算次数. 特别地, 考虑 $m=n=p$ 和 $m=n, p=1$ 情况.

9. 设 $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ 都是长度为 n 的列向量. 由矩阵乘法的法则, $(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v})\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{v}\boldsymbol{w}^{\mathrm{T}})$, 记 $\boldsymbol{x} = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{v}\boldsymbol{w}^{\mathrm{T}}$. 求这两种计算 $\boldsymbol{x}$ 方法的运算次数. 从花费计算量角度看, 哪一个方法更好?

10. 设 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ 分别为 $m\times n$, $n\times p$, $p\times q$ 阶矩阵. 求计算 $(\boldsymbol{AB})\boldsymbol{C}$ 和 $\boldsymbol{A}(\boldsymbol{BC})$ 中包含的乘法运算次数. 对于 m, n, p, q 的特殊取值, 证明这两种计算方法的运算次数可能差别很大.

11. 定义 $\boldsymbol{B} = \boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}$, $\boldsymbol{w}$ 是长度为 n 的列向量.

(a) 给出形成 $\boldsymbol{B}$ 的运算次数. 尽可能做到高效.

(b) 设 $\boldsymbol{A}$ 为任意 $n\times n$ 阶矩阵. 利用 (a) 中形成的矩阵 $\boldsymbol{B}$, 给出形成乘积 $\boldsymbol{AB}$ 所需的运算次数.

(c) 给出另一种计算乘积 $\boldsymbol{AB} = \boldsymbol{A}\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}$ 运算次数较少的方法.

12. 设 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 是长度为 n 的列向量. 定义 $\boldsymbol{A} = \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$, 注意 $\boldsymbol{A}$ 为 $n\times n$ 阶方阵.

(a) 给出一般元素 $\boldsymbol{A}_{i,j}$. 在计算机中由 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 得到 $\boldsymbol{A}$ 的运算次数是多少?

(b) 给出计算 $\boldsymbol{A}\boldsymbol{x}$ 的运算次数, 这里 $\boldsymbol{x}$ 是长度为 n 的一般列向量.

(c) 给出另一种计算 $\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}})\boldsymbol{x}$ 运算次数较少的方法.

13. 用初等行运算将下列矩阵化为恒等矩阵:

(a) $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ (b) $\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$

14. 用初等行运算将下列矩阵化为下列形式:

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & x \end{bmatrix}$$

这里 x 表示某个任意数.

(a) $\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 4 & 1 \\ 9 & 35 & 7 \end{bmatrix}$

15. 给出满足 $\boldsymbol{AB} = \boldsymbol{0}$ 的两个 2 阶方阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$, 使得 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的所有元素非零. 能说两个矩阵中有一个是非奇异矩阵吗?

16. 设 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 是使乘积 $\boldsymbol{AB}$ 奇异的两个方阵. 证明 $\boldsymbol{A}$ 或 $\boldsymbol{B}$ 是奇异的.

17. 为了使

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

和

$$\boldsymbol{B} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

可交换, a, b, c, d 必须满足什么条件?

18. 设 $\boldsymbol{A}$ 是一个方阵. 证明

$$(\boldsymbol{I} + 2\boldsymbol{A} + 3\boldsymbol{A}^2)(2\boldsymbol{I} - \boldsymbol{A}) = (2\boldsymbol{I} - \boldsymbol{A})(\boldsymbol{I} + 2\boldsymbol{A} + 3\boldsymbol{A}^2)$$

这样, 矩阵 $\boldsymbol{I}+2\boldsymbol{A}+3\boldsymbol{A}^2$ 和 $2\boldsymbol{I}-\boldsymbol{A}$ 可交换.

注记: 设

$$p(t)=a_0+a_1t+\cdots+a_kt^k$$
$$q(t)=b_0+b_1t+\cdots+b_lt^l$$

是 k 次和 l 次多项式. 定义矩阵

$$p(\boldsymbol{A})=a_0\boldsymbol{I}+a_1\boldsymbol{A}+\cdots+a_k\boldsymbol{A}^k$$

$$q(\boldsymbol{A})=b_0\boldsymbol{I}+b_1\boldsymbol{A}+\cdots+b_l\boldsymbol{A}^l$$

可以证明 $p(\boldsymbol{A})q(\boldsymbol{A})=q(\boldsymbol{A})p(\boldsymbol{A})$.

19. 用习题 18 中介绍的多项式矩阵记号, 设

$$p(t)=(t-r_1)(t-r_2)=t^2-t(r_1+r_2)+r_1r_2$$

对任意方阵 $\boldsymbol{A}$, 证明

$$p(\boldsymbol{A})=(\boldsymbol{A}-r_1\boldsymbol{I})(\boldsymbol{A}-r_2\boldsymbol{I})=\boldsymbol{A}^2-\boldsymbol{A}(r_1+r_2)+r_1r_2\boldsymbol{I}$$

20. 对 2 阶矩阵的逆阵导出公式 (6.20).

21. 检验 $n\times n$ 矩阵

$$\boldsymbol{A}=\begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

的逆是

$$\boldsymbol{A}^{-1}=\begin{bmatrix} n & n-1 & n-2 & n-3 & \cdots & 1 \\ n-1 & n-1 & n-2 & n-3 & \cdots & 1 \\ n-2 & n-2 & n-2 & n-3 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

22. 设 $\boldsymbol{A}$ 为 $m\times n$ 阶, $\boldsymbol{B}$ 为 $n\times p$ 阶. 证明 $(\boldsymbol{AB})^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}$.

提示: 模仿证明 (6.23), $(\boldsymbol{AB})\boldsymbol{C}=\boldsymbol{A}(\boldsymbol{BC})$ 时使用过的方法.

23. 设 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是同阶非奇异矩阵. 证明 $\boldsymbol{AB}$ 非奇异, 并且 $(\boldsymbol{AB})^{-1}=\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$.

提示: 用 $\boldsymbol{AB}$ 相乘.

24. 证明 $(\boldsymbol{A}^{\mathrm{T}})^{-1}=(\boldsymbol{A}^{-1})^{\mathrm{T}}$.

25. 对 2 阶线性方程组, 证明定理 6.2.10 中 (S1) 和 (S4) 的等价性.

26. 回顾 6.1 节习题 1 中所给的三次多项式插值的重组以后的问题. 为证明关于 a_0, a_1, a_2, a_3 的线性方程组总有唯一解, 证明定理 6.2.10 中相应的性质 (S3) 成立.

27. 考虑通过定理 6.2.10 中的 (S3) 来证明 6.1 节中习题 4 的方程组是非奇异的. 设 $\boldsymbol{x}$ 是齐次方程组的解, 并设

$$\alpha = \max_{1 \leqslant i \leqslant n} |x_i|$$

至少在一个下标处取得最大值, 比如说 $\alpha = |x_k|$. 观察第 k 个方程, 利用它, 可以证明对 α 只有 $\alpha = 0$ 一种可能, 从而证明了对所有 i, $x_i = 0$. 分别考虑 $k = 1$ 或 $k = n$ 以及 $1 < k < n$ 的情况.

28. 利用定理 6.2.6 并假设 $\det(\boldsymbol{A}) = 0$, 则至少有一个向量 $\boldsymbol{x} \neq \boldsymbol{0}$ 满足 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. 对奇异矩阵

$$\begin{bmatrix} 0 & 4 & 1 & 1 \\ 4 & 0 & 1 & 1 \\ 1 & 1 & -1 & 2 \\ 1 & 1 & 2 & -1 \end{bmatrix}$$

找一个这样的 $\boldsymbol{x}$.

29. (a) 对于

$$\det[\lambda \boldsymbol{I} - \boldsymbol{B}] = 0, \quad \boldsymbol{B} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

求 λ 值. 称 λ 为矩阵 $\boldsymbol{B}$ 的本征值, 参看 7.2 节.

(b) 用 (a) 中得到的 λ 值, 对于 $\boldsymbol{A} = \lambda \boldsymbol{I} - \boldsymbol{B}$, 找出一个 $\boldsymbol{x} \neq \boldsymbol{0}$ 满足, $(\lambda \boldsymbol{I} - \boldsymbol{B})\boldsymbol{x} = \boldsymbol{0}$ 或等价地 $\boldsymbol{B}\boldsymbol{x} = \lambda \boldsymbol{x}$. 称这样的一个向量 $\boldsymbol{x}$ 为对应于本征值 λ 的本征向量.

30. 考虑下列矩阵, 称之为初等矩阵. 设 $\boldsymbol{I}$ 是 n 阶恒等矩阵, 对某个 $1 \leqslant k \leqslant n-1$, 修改对角线下第 k 列. 定义

$$\boldsymbol{E}_k = \begin{bmatrix} 1 & 0 & \cdots & 0 & & \cdots & & & 0 \\ 0 & 1 & & 0 & & & & & \\ \vdots & & \ddots & \vdots & & & & & \\ & & & 1 & 0 & & & & \\ & & & 0 & 1 & \ddots & & & \vdots \\ & & & 0 & a_{k+1} & 1 & & & \\ & & & 0 & a_{k+2} & 0 & \ddots & & \\ \vdots & & & \vdots & \vdots & \vdots & & 1 & 0 \\ 0 & 0 & \cdots & 0 & a_n & 0 & \cdots & 0 & 1 \end{bmatrix}$$

(a) 证明初等矩阵之积是下三角矩阵.

(b) 如果 $\boldsymbol{A}$ 为 $n \times m$ 阶矩阵, 证明 $\boldsymbol{E}_k\boldsymbol{A}$ 可如下形成: 对于 $k+1 \leqslant i \leqslant n$, $\boldsymbol{A}$ 的第 k 行的 a_i 倍加到 $\boldsymbol{A}$ 的第 i 行, 形成 $\boldsymbol{E}_k\boldsymbol{A}$ 的新的第 i 行, $\boldsymbol{A}$ 的第 $1, 2, \cdots, k$ 行保持不变.

(c) 计算 $\boldsymbol{E}_k$ 的逆.

提示: 利用 (b) 的结论, 反向进行形成 $\boldsymbol{E}_k\boldsymbol{A}$ 的运算.

6.3　高斯消元法

为求解大型线性方程组, 我们利用初等代数课程中所教过的方法. 这里先介绍用这种方法求解特殊 3 阶线性方程组. 考虑方程组

$$\begin{cases} x_1+2x_2+\ x_3=0 & \mathrm{E}(1)\\ 2x_1+2x_2+3x_3=3 & \mathrm{E}(2)\\ -x_1-3x_2\qquad\ =2 & \mathrm{E}(3)\end{cases} \tag{6.35}$$

为了更方便引用, 我们对每个方程做上标记.

第 1 步　从 E(2) 和 E(3) 消去 x_1. 从 E(2) 减去 E(1) 的 2 倍, 并且从 E(3) 减去 E(1) 的 (−1) 倍. 得到新的线性方程组:

$$\begin{cases} x_1+2x_2+x_3=0 & \mathrm{E}(1)\\ \quad -2x_2+x_3=3 & \mathrm{E}(2)\\ \quad\ \ -x_2+x_3=2 & \mathrm{E}(3)\end{cases}$$

第 2 步　从 E(3) 消去 x_2. 从 E(3) 减去 E(2) 的 $\frac{1}{2}$ 倍, 得到

$$\begin{cases} x_1+2x_2+x_3=0 & \mathrm{E}(1)\\ \quad -2x_2+x_3=3 & \mathrm{E}(2)\\ \qquad\quad \frac{1}{2}x_3=\frac{1}{2} & \mathrm{E}(3)\end{cases} \tag{6.36}$$

第 3 步　逐次解出 x_3, x_2, x_1.

$$\begin{aligned} x_3&=1\\ -2x_2+1&=3\\ x_2&=-1\\ x_1+2(-1)+1&=0\\ x_1&=1\end{aligned}$$

第 1 步和第 2 步是**消元** (elimination) 步, 得到的结果 (6.36) 是一个**上三角线性方程组**. 这个方程组与 (6.35) 同解, 但 (6.36) 更容易求解. 第 3 步称为**回代** (back substitution) 求解. 整个过程称为**高斯消元法**, 一般对小型至中型**线性方程组**, 比如说几百阶的, 它是最有效的求解方法.

消元步骤用矩阵记号和运算执行更方便. 为此, 对方程组 (6.35) 我们用增广矩阵形式:

$$[\boldsymbol{A}|\boldsymbol{b}]=\left[\begin{array}{rrr|r} 1 & 2 & 1 & 0\\ 2 & 2 & 3 & 3\\ -1 & -3 & 0 & 2\end{array}\right] \tag{6.37}$$

注意到从一个方程组中消去一个变量等价于通过初等行运算使相应系数化为零. 例如, 为从 E(2) 和 E(3) 消去 x_1, 我们将第 1 行乘以 (-2) 再加到第 2 行, 然后, 将第 1 行加到第 3 行, 得到

$$\left[\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 0 & -2 & 1 & 3 \\ 0 & -1 & 1 & 2 \end{array}\right]$$

随后, 为从 E(3) 消去 x_2, 将第 2 行乘以 $(-1/2)$ 再加到第 3 行:

$$\left[\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 0 & -2 & 1 & 3 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{array}\right]$$

这个矩阵是从矩阵 (6.37) 通过初等行运算得到的, 并且已把原方程组 (6.35) 化简为与 (6.36) 一样的上三角方程组

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ \frac{1}{2} \end{bmatrix} \tag{6.38}$$

然后, 用回代过程求解 (6.38).

在初等代数课程中, 常常用有点随机应变的方式消去变量. 一种教过的方式是在消元中找一个简单的整数系数用于消元, 为了使分数的计算量最小. 但在计算机上, 处理分数并不比用整数更困难. 为编写计算机程序, 算法必须有一个明确给定的做法, 它不依赖是否系数碰巧是一个整数.

为了观察高斯消元法的形式结构, 我们再看 $n=3$ 时具有一般系数的情况. 求解方程组是

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 & \text{E(1)} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 & \text{E(2)} \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 & \text{E(3)} \end{cases} \tag{6.39}$$

第 1 步 从 E(2) 和 E(3) 消去 x_1. 为简化描述, 假设 $a_{11} \neq 0$; 在本节后面将去掉这个假设. 定义

$$m_{21} = \frac{a_{21}}{a_{11}}, \quad m_{31} = \frac{a_{31}}{a_{11}}$$

从 E(2) 减去 E(1) 的 m_{21} 倍, 再从 E(3) 减去 E(1) 的 m_{31} 倍. 这样就把 (6.39) 化

为等价的方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 & \text{E}(1) \\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 = b_2^{(2)} & \text{E}(2) \\ a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 = b_3^{(2)} & \text{E}(3) \end{cases} \tag{6.40}$$

其中

$$a_{ij}^{(2)} = a_{ij} - m_{i1}a_{1j}, \quad i,j = 2,3$$

$$b_i^{(2)} = b_i - m_{i1}b_1, \quad i = 2,3$$

第 2 步　从 E(3) 消去 x_2. 暂时再假设 $a_{22}^{(2)} \neq 0$. 定义

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}}$$

从 E(3) 减去 E(2) 的 m_{32} 倍. 得到

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 & \text{E}(1) \\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 = b_2^{(2)} & \text{E}(2) \\ a_{33}^{(3)}x_3 = b_3^{(3)} & \text{E}(3) \end{cases}$$

其中

$$a_{33}^{(3)} = a_{33}^{(2)} - m_{32}a_{23}^{(2)}, \quad b_3^{(3)} = b_3^{(2)} - m_{32}b_2^{(2)}$$

第 3 步　用回代逐次解出 x_3, x_2, x_1.

$$\begin{aligned} x_3 &= \frac{b_3^{(3)}}{a_{33}^{(3)}} \\ x_2 &= \frac{b_2^{(2)} - a_{23}^{(2)}x_3}{a_{22}^{(2)}} \\ x_1 &= \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}} \end{aligned} \tag{6.41}$$

容易将这个 $n = 3$ 的算法推广到一般的 n 个线性方程的非奇异方程组, 当然得假设某些元素非零, 但这个假设在本节后面将去掉. 为了描述的方便, 将原始方程组表示为

$$\begin{cases} a_{11}^{(1)}x_1 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} & \text{E}(1) \\ \qquad\vdots & \\ a_{n1}^{(1)}x_1 + \cdots + a_{nn}^{(1)}x_n = b_n^{(1)} & \text{E}(n) \end{cases} \tag{6.42}$$

对 $k=1,2,\cdots,n-1$, 执行下列消元步骤.

第 k 步 从 $\mathrm{E}(k+1)$ 到 $\mathrm{E}(n)$ 中消去 x_k. 从前面第 $1,\cdots,k-1$ 步已得到下列形式的方程组

$$\left\{\begin{array}{llll} a_{11}^{(1)}x_1+a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} & \mathrm{E}(1) \\ \qquad a_{22}^{(2)}x_2 + \cdots + a_{2n}^{(2)}x_n = b_2^{(2)} & \mathrm{E}(2) \\ \qquad\qquad \ddots \qquad \vdots & \\ \qquad\qquad a_{kk}^{(k)}x_k+\cdots+a_{kn}^{(k)}x_n = b_k^{(k)} & \mathrm{E}(k) \\ \qquad\qquad\qquad \vdots & \\ \qquad\qquad a_{nk}^{(k)}x_k+\cdots+a_{nn}^{(k)}x_n = b_n^{(k)} & \mathrm{E}(n) \end{array}\right. \tag{6.43}$$

假设 $a_{kk}^{(k)}\neq 0$, 并定义乘数

$$m_{ik}=\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}},\quad i=k+1,\cdots,n \tag{6.44}$$

对于方程 $i=k+1,\cdots,n$, 从 $\mathrm{E}(i)$ 减去 $\mathrm{E}(k)$ 的 m_{ik} 倍, 这样就从 $\mathrm{E}(i)$ 消去了 x_k. $\mathrm{E}(k+1)$ 到 $\mathrm{E}(n)$ 中新的系数和右端定义为

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)}-m_{ik}a_{kj}^{(k)},\quad i,j=k+1,\cdots,n \\ b_i^{(k+1)} &= b_i^{(k)}-m_{ik}b_k^{(k)},\quad i=k+1,\cdots,n \end{aligned} \tag{6.45}$$

当完成第 $n-1$ 步后, 线性方程组成为上三角形式. 我们将它表示为

$$\left\{\begin{aligned} u_{11}x_1+\cdots+u_{1n}x_n &= g_1 \\ &\vdots \\ u_{nn}x_n &= g_n \end{aligned}\right. \tag{6.46}$$

这些系数与前面系数的关系为

$$u_{ij}=a_{ij}^{(i)},\quad g_i=b_i^{(i)}$$

第 n 步 用回代求解 (6.46). 在 (6.46) 中逐次求解 $x_n,x_{n-1},\cdots,x_1$.

$$\begin{aligned} x_n &= \frac{g_n}{u_{nn}} \\ x_i &= \frac{g_i-\sum\limits_{j=i+1}^{n}u_{ij}x_j}{u_{ii}},\quad i=n-1,\cdots,1 \end{aligned} \tag{6.47}$$

这完成了高斯消元法的定义.

注意, 每步消元过程都可以通过对增广矩阵 $[\boldsymbol{A}|\boldsymbol{b}]$ 的某一行进行初等行运算来实现, 如同在本节开始部分例子中给出的那样.

例 6.3.1 求解方程组

$$\left\{\begin{aligned} 4x_1+3x_2+2x_3+\ \ x_4&=\ \ 1\\ 3x_1+4x_2+3x_3+2x_4&=\ \ 1\\ 2x_1+3x_2+4x_3+3x_4&=-1\\ x_1+2x_2+3x_3+4x_4&=-1 \end{aligned}\right. \tag{6.48}$$

首先建立增广矩阵, 然后给出对这个方程组进行高斯消元法的第 1 步、第 2 步和第 3 步的结果.

$$\left[\begin{array}{cccc|c} 4 & 3 & 2 & 1 & 1\\ 3 & 4 & 3 & 2 & 1\\ 2 & 3 & 4 & 3 & -1\\ 1 & 2 & 3 & 4 & -1 \end{array}\right] \begin{array}{r} m_{21}=\frac{3}{4}\\ \longrightarrow\\ m_{31}=\frac{1}{2}\\ m_{41}=\frac{1}{4} \end{array} \left[\begin{array}{cccc|c} 4 & 3 & 2 & 1 & 1\\ 0 & \frac{7}{4} & \frac{3}{2} & \frac{5}{4} & \frac{1}{4}\\ 0 & \frac{3}{2} & 3 & \frac{5}{2} & -\frac{3}{2}\\ 0 & \frac{5}{4} & \frac{5}{2} & \frac{15}{4} & -\frac{5}{4} \end{array}\right]$$

$$m_{32}=\frac{6}{7}\downarrow m_{42}=\frac{5}{7}$$

$$\left[\begin{array}{cccc|c} 4 & 3 & 2 & 1 & 1\\ 0 & \frac{7}{4} & \frac{3}{2} & \frac{5}{4} & \frac{1}{4}\\ 0 & 0 & \frac{12}{7} & \frac{10}{7} & -\frac{12}{7}\\ 0 & 0 & 0 & \frac{5}{3} & 0 \end{array}\right] \overleftarrow{m_{43}}=\frac{5}{6} \left[\begin{array}{cccc|c} 4 & 3 & 2 & 1 & 1\\ 0 & \frac{7}{4} & \frac{3}{2} & \frac{5}{4} & \frac{1}{4}\\ 0 & 0 & \frac{12}{7} & \frac{10}{7} & -\frac{12}{7}\\ 0 & 0 & \frac{10}{7} & \frac{20}{7} & -\frac{10}{7} \end{array}\right]$$

用回代逐次求解 x_4, x_3, x_2, x_1, 得到方程组的解

$$x_1=0,\quad x_2=1,\quad x_3=-1,\quad x_4=0$$

■

6.3.1 部分选主元

在消元过程中, 我们假设了对于 $k=1,2,\cdots,n-1$ 的每一步 $a_{kk}^{(k)}\neq 0$. 当 $a_{kk}^{(k)}=0$ 时除去假设, 我们来检查 (6.43) 式中 E(k) 下面的方程. 由于假设原方程组 (6.42) 非奇异, 故可以证明 E(k) 下面的方程中必有一个方程其 x_k 项的系数非零 (见习题 4). 将这个方程与 E(k) 交换得到一个新的等价方程组就能使 $a_{kk}^{(k)}\neq 0$ 成立. 但在数值计算中, $a_{kk}^{(k)}$ 非零不是充分的, 如下例说明的那样.

例 6.3.2 求解

$$\left\{\begin{aligned} 6x_1+2x_2+2x_3&=-2\\ 2x_1+\tfrac{2}{3}x_2+\tfrac{1}{3}x_3&=\ \ 1\\ x_1+2x_2-\ \ x_3&=\ \ 0 \end{aligned}\right. \tag{6.49}$$

这个方程组的解为

$$x_1 = 2.6, \quad x_2 = -3.8, \quad x_3 = -5.0 \tag{6.50}$$

我们使用十进制计算机求解这个方程组. 设计算机用 4 位有效数字浮点表示, 并且所有运算过程将四舍五入. 用增广矩阵表示消元步骤, 从 (6.49) 的机器表示开始.

$$\left[\begin{array}{ccc|c} 6.000 & 2.000 & 2.000 & -2.000 \\ 2.000 & 0.666\,7 & 0.333\,3 & 1.000 \\ 1.000 & 2.000 & -1.000 & 0.0 \end{array}\right]$$

$$\downarrow m_{21} = 0.333\,3, \quad m_{31} = 0.166\,7$$

$$\left[\begin{array}{ccc|c} 6.000 & 2.000 & 2.000 & -2.000 \\ 0.0 & 0.000\,100\,0 & -0.333\,3 & 1.667 \\ 0.0 & 1.667 & -1.333 & 0.333\,4 \end{array}\right] \tag{6.51}$$

$$\downarrow m_{32} = 16\,670$$

$$\left[\begin{array}{ccc|c} 6.000 & 2.000 & 2.000 & -2.000 \\ 0.0 & 0.000\,100\,0 & -0.333\,3 & 1.667 \\ 0.0 & 0.0 & 5\,555 & -27\,790 \end{array}\right]$$

利用回代, 我们得到下列近似解:

$$x_1 = 1.335, \quad x_2 = 0.0, \quad x_3 = -5.003$$

应将它与 (6.50) 给出的真解比较.

这个消元过程困难在 (6.51) 式中, 第 2 行第 2 列元素应该为零, 但舍入误差妨碍了它为零. 这意味着在 (2,2) 位置的系数本质上有无限大的相对误差, 这个误差贯穿于包含这个系数的计算中. 为了避免这种情况, 我们交换 (6.51) 式中第 2 行和第 3 行, 然后继续消元. 最后的矩阵为

$$\left[\begin{array}{ccc|c} 6.000 & 2.000 & 2.000 & -2.000 \\ 0.0 & 1.667 & -1.333 & 0.333\,4 \\ 0.0 & 0.0 & -0.333\,2 & 1.667 \end{array}\right]$$

最后的乘数是 $m_{32} = 0.000\,059\,99$. 用回代, 得到近似解

$$x_1 = 2.602, \quad x_2 = -3.801, \quad x_3 = -5.003$$

这个解与 (6.50) 给出的真解很接近. ■

为了避免在这个例子中前半部分出现的问题, 我们使用下列策略. 在第 k 步 [见 (6.43)], 计算

$$c = \max_{k \leqslant i \leqslant n} \left| a_{ik}^{(k)} \right|$$

这是第 k 步系数矩阵第 k 列中第 k 行向下到最后一行的元素的最大绝对值. 如果元素 $\left| a_{kk}^{(k)} \right| < c$, 那么交换 $\mathrm{E}(k)$ 与它下面的一个方程, 得到一个新的方程 $\mathrm{E}(k)$, 使 $\left| a_{kk}^{(k)} \right| = c$. 这样做使 $a_{kk}^{(k)}$ 尽可能远离零. 元素 $a_{kk}^{(k)}$ 称为第 k 步消元过程的**主元** (pivot element), 在这段中描述的过程称为部分选主元 (partial pivoting), 或简称选主元. 还有其他选主元的形式, 但部分选主元是实践中最流行的一种.

这里介绍的部分选主元避免使用几乎为零的系数作主元. 此外, 使用选主元有另一个同样重要的原因: 在大多数例子中, 它减少舍入误差传播的影响. 用部分选主元, (6.44) 中定义的乘数 m_{ik} 满足

$$|m_{ik}| \leqslant 1, \quad 1 \leqslant k < i \leqslant n$$

因为乘以 m_{ik} 不会导致产生更大的数, 所以这将有助于减少有效数字损失造成的误差.

例 6.3.3 利用在上例中描述的 4 位十进制小数计算机求解方程组

$$\begin{cases} 0.729x_1 + 0.81x_2 + 0.9x_3 = 0.686\,7 \\ \qquad x_1 + \qquad x_2 + \qquad x_3 = 0.833\,8 \\ 1.331x_1 + 1.21x_2 + 1.1x_3 = 1.000 \end{cases}$$

舍入到 4 位小数的精确解为

$$x_1 = 0.224\,5, \quad x_2 = 0.281\,4, \quad x_3 = 0.327\,9 \tag{6.52}$$

不选主元的解

$$\left[\begin{array}{ccc|c} 0.729\,0 & 0.810\,0 & 0.900\,0 & 0.686\,7 \\ 1.000 & 1.000 & 1.000 & 0.833\,8 \\ 1.331 & 1.210 & 1.100 & 1.000 \end{array}\right]$$

$$\downarrow \begin{array}{l} m_{21} = 1.372 \\ m_{31} = 1.826 \end{array}$$

$$\left[\begin{array}{ccc|c} 0.729\,0 & 0.810\,0 & 0.900\,0 & 0.686\,7 \\ 0.0 & -0.111\,0 & -0.235\,0 & -0.108\,4 \\ 0.0 & -0.269\,0 & -0.543\,0 & -0.254\,0 \end{array}\right]$$

$$\downarrow \; m_{32} = 2.423$$

$$\left[\begin{array}{ccc|c} 0.729\,0 & 0.810\,0 & 0.900\,0 & 0.686\,7 \\ 0.0 & -0.111\,0 & -0.235\,0 & -0.108\,4 \\ 0.0 & 0.0 & 0.026\,40 & 0.008\,700 \end{array}\right]$$

这样, 方程组的解为

$$x_1 = 0.225\,1, \quad x_2 = 0.279\,0, \quad x_3 = 0.329\,5 \tag{6.53}$$

选主元的解

我们使用记号 $r_i \leftrightarrow r_j$ 表示交换第 i 和第 j 行.

$$\left[\begin{array}{ccc|c} 0.729\,0 & 0.810\,0 & 0.900\,0 & 0.686\,7 \\ 1.000 & 1.000 & 1.000 & 0.833\,8 \\ 1.331 & 1.210 & 1.100 & 1.000 \end{array}\right]$$

$$r_1 \leftrightarrow r_3 \downarrow \begin{array}{l} m_{21} = 0.751\,3 \\ m_{31} = 0.547\,7 \end{array}$$

$$\left[\begin{array}{ccc|c} 1.331 & 1.210 & 1.100 & 1.000 \\ 0.0 & 0.090\,90 & 0.173\,6 & 0.082\,50 \\ 0.0 & 0.147\,3 & 0.297\,5 & 0.139\,0 \end{array}\right]$$

$$r_2 \leftrightarrow r_3 \downarrow m_{32} = 0.617\,1$$

$$\left[\begin{array}{ccc|c} 1.331 & 1.210 & 1.100 & 1.000 \\ 0.0 & 0.147\,3 & 0.297\,5 & 0.139\,0 \\ 0.0 & 0.0 & -0.010\,00 & -0.003\,280 \end{array}\right]$$

这样, 方程组的解为

$$x_1 = 0.224\,6, \quad x_2 = 0.281\,2, \quad x_3 = 0.328\,0 \tag{6.54}$$

把这个结果与 (6.52) 比较, 我们观察到这是一个比 (6.53) 更精确的解. ■

6.3.2 逆矩阵的计算

我们可以利用高斯消元法来计算 $\boldsymbol{A}^{-1}$. 为了说明这种方法, 考虑 3 阶矩阵. 用 $\boldsymbol{X}$ 表示未知逆阵, $\boldsymbol{X} = \boldsymbol{A}^{-1}$, 并用 $\boldsymbol{x}_{*1}, \boldsymbol{x}_{*2}, \boldsymbol{x}_{*3}$ 表示 $\boldsymbol{X}$ 的 3 列. 同样, 用 $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ 表示 $\boldsymbol{I}_3$ 的 3 列

$$\boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

等式

$$\boldsymbol{AX}=\boldsymbol{I}$$

成为

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.55}$$

仅考虑乘积的第一列

$$\begin{bmatrix} a_{11}x_{11}+a_{12}x_{21}+a_{13}x_{31} \\ a_{21}x_{11}+a_{22}x_{21}+a_{23}x_{31} \\ a_{31}x_{11}+a_{32}x_{21}+a_{33}x_{31} \end{bmatrix}=\boldsymbol{Ax}_{*1}$$

类似地, $\boldsymbol{Ax}_{*2}$ 和 $\boldsymbol{Ax}_{*3}$ 是 $\boldsymbol{AX}$ 的第 2 列和第 3 列. 于是, (6.55) 可以写成

$$\boldsymbol{AX}=\begin{bmatrix} \boldsymbol{Ax}_{*1}, & \boldsymbol{Ax}_{*2}, & \boldsymbol{Ax}_{*3} \end{bmatrix}=\begin{bmatrix} \boldsymbol{e}_1, & \boldsymbol{e}_2, & \boldsymbol{e}_3 \end{bmatrix}$$

通过匹配相应的列, 得到

$$\boldsymbol{Ax}_{*1}=\boldsymbol{e}_1, \quad \boldsymbol{Ax}_{*2}=\boldsymbol{e}_2, \quad \boldsymbol{Ax}_{*3}=\boldsymbol{e}_3 \tag{6.56}$$

因此, $\boldsymbol{X}$ 的列是具有相同系数矩阵 $\boldsymbol{A}$ 的 3 个联立线性方程组的解. 可以用高斯消元法求解这些方程组.

例 6.3.4 求

$$\boldsymbol{A}=\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix}$$

的逆. 对增广矩阵

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 1 & 2 & -2 & 0 & 1 & 0 \\ -2 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

执行高斯消元法. 我们同时解 (6.56) 中的 3 个方程组. 消元步骤是

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 3 & -1 & 2 & 0 & 1 \end{array}\right] \quad \begin{aligned} m_{21}&=1 \\ m_{31}&=-2 \end{aligned}$$

$$\downarrow$$

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 2 & 5 & -3 & 1 \end{array}\right] \quad m_{32}=3$$

我们解下列方程组来计算 x_{*1}

$$\begin{cases} x_{11}+x_{21}-x_{31}=1 \\ x_{21}-x_{31}=-1 \\ 2x_{31}=5 \end{cases}$$

用回代求解, 得到方程组的解为

$$x_{11}=2, \quad x_{21}=\frac{3}{2}, \quad x_{31}=\frac{5}{2}$$

类似地, 可以求得 $\boldsymbol{X}$ 的第 2 列和第 3 列. 这样,

$$\boldsymbol{X}=\boldsymbol{A}^{-1}=\begin{bmatrix} 2 & -1 & 0 \\ \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{5}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \tag{6.57}$$

可以验算, 将这个矩阵乘以 $\boldsymbol{A}$ 得到恒等矩阵.

我们还可以对三角形方程组继续进行消元来的得到 $\boldsymbol{X}$. 也就是说, 我们继续应用初等行运算直到增广矩阵的左边一半变为一个恒等矩阵.

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 2 & 5 & -3 & 1 \end{array}\right]$$

↓第 3 行除以 2

$$\left[\begin{array}{ccc|ccc} 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 1 & \frac{5}{2} & -\frac{3}{2} & \frac{1}{2} \end{array}\right]$$

↓第 3 行加到第 1 行与第 2 行

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 0 & \frac{7}{2} & -\frac{3}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 & \frac{5}{2} & -\frac{3}{2} & \frac{1}{2} \end{array}\right]$$

↓第 1 行减去第 2 行

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 1 & 0 & \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 & \frac{5}{2} & -\frac{3}{2} & \frac{1}{2} \end{array}\right]$$

这样, 原方程组

$$\boldsymbol{AX} = \boldsymbol{I}$$

转变为

$$\boldsymbol{IX} = \begin{bmatrix} 2 & -1 & 0 \\ \frac{3}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{5}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix}$$

所以, (6.57) 是 $\boldsymbol{A}$ 的逆阵. ■

上面的过程可以推广到任意 $n \geqslant 1$ 阶的矩阵 $\boldsymbol{A}$. 用与 $\boldsymbol{A}$ 同阶的恒等矩阵建立 $\boldsymbol{A}$ 的增广矩阵, 再应用上面所述的高斯消元法.

6.3.3 运算量

知道计算的运算量是重要的. 为此, 我们分三个部分来计算高斯消元法中涉及的算术运算次数. 为方便起见, 记 $\boldsymbol{U}$ 和 $\boldsymbol{g}$ 表示 (6.46) 式中的系数矩阵和右端向量.

1. 消元步骤. 计算从方程组 (6.42) 化到三角形方程组 (6.46) 所做加法 (减法)、乘法和除法的次数. 我们仅考虑 (6.42) 式中系数矩阵 $\boldsymbol{A}$ 的运算量而不考虑右端的 $\boldsymbol{b}$. 表 6-1 包含消元每一步中这些运算的次数. 得到表中最后一行合计是利用了公式

$$\sum_{j=1}^{p} j = \frac{p(p+1)}{2}, \quad \sum_{j=1}^{p} j^2 = \frac{p(p+1)(2p+1)}{6}, \quad p \geqslant 1$$

由于除法和乘法的运算时间差不多, 它们的运算量合在一起计算. 这样我们得到

$$AS(\boldsymbol{A} \to \boldsymbol{U}) = \frac{n(n-1)(2n-1)}{6}$$

$$MD(\boldsymbol{A} \to \boldsymbol{U}) = \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} = \frac{n(n^2-1)}{3}$$

$AS(\cdot)$ 表示所附运算的加法和减法次数, $MD(\cdot)$ 表示所附运算的乘法和除法次数. 记号 $\boldsymbol{A} \to \boldsymbol{U}$ 表示将 (6.42) 中 $\boldsymbol{A}$ 化到 (6.46) 中 $\boldsymbol{U}$ 的消元过程.

表 6-1 $\boldsymbol{A}$ 化为 $\boldsymbol{U}$ 的运算量

步骤	加法	乘法	除法
1	$(n-1)^2$	$(n-1)^2$	$n-1$
2	$(n-2)^2$	$(n-2)^2$	$n-2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	1	1	1
总计	$\frac{n(n-1)(2n-1)}{6}$	$\frac{n(n-1)(2n-1)}{6}$	$\frac{n(n-1)}{2}$

2. **右端的 $\boldsymbol{b}$ 修正为 $\boldsymbol{g}$.** 与上面类似, 可以得到

$$AS(\boldsymbol{b}\to\boldsymbol{g})=(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}$$

$$MD(\boldsymbol{b}\to\boldsymbol{g})=(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}$$

3. **回代步骤, 由 (6.46) 求 $\boldsymbol{x}$.** 如上面一样,

$$AS(\boldsymbol{g}\to\boldsymbol{x})=0+1+\cdots+(n-1)=\frac{n(n-1)}{2}$$
$$MD(\boldsymbol{g}\to\boldsymbol{x})=1+2+\cdots+n=\frac{n(n+1)}{2}$$

将这些结果合在一起, 我们得到计算 $\boldsymbol{x}$ 所需的全部运算次数为

$$\begin{aligned}AS(\boldsymbol{x})&=AS(\boldsymbol{A}\to\boldsymbol{U})+AS(\boldsymbol{b}\to\boldsymbol{g})+AS(\boldsymbol{g}\to\boldsymbol{x})\\&=\frac{n(n-1)(2n-1)}{6}+\frac{n(n-1)}{2}+\frac{n(n-1)}{2}\\&=\frac{n(n-1)(2n+5)}{6}\end{aligned}$$

$$MD(\boldsymbol{x})=\frac{n(n^2+3n-1)}{3}$$

由于这些运算量中 AS 与 MD 几乎都相同, 因此只对 MD 进行讨论. n 较大时, 高斯消元法运算量大约为 $\frac{1}{3}n^3$. 这意味当 n 加倍时, 求解线性方程组的运算量增至 8 倍. 另外, 由于余下步骤所需的运算量为

$$MD(\boldsymbol{b}\to\boldsymbol{g})+MD(\boldsymbol{g}\to\boldsymbol{x})=\frac{n(n-1)}{2}+\frac{n(n+1)}{2}=n^2 \tag{6.58}$$

故高斯消元法的运算量大部分是在 $\boldsymbol{A}\to\boldsymbol{U}$ 消元步骤. 因此, 一旦 $\boldsymbol{A}\to\boldsymbol{U}$ 步完成, 接下来求解线性方程组的运算量就少得多. 我们将在 6.5 节中, 给出一个简单易行的方法估计用高斯消元法求得解的误差.

考虑求解线性方程组

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$$

这里, $\boldsymbol{A}$ 为 n 阶非奇异矩阵. 则 $\boldsymbol{A}^{-1}$ 存在, 并且

$$\begin{aligned}\boldsymbol{A}^{-1}(\boldsymbol{A}\boldsymbol{x})&=\boldsymbol{A}^{-1}\boldsymbol{b}\\\boldsymbol{x}&=\boldsymbol{A}^{-1}\boldsymbol{b}\end{aligned} \tag{6.59}$$

因此, 如果 $\boldsymbol{A}^{-1}$ 已知, 则用矩阵乘法可以求得 $\boldsymbol{x}$.

由此, 先求 $\boldsymbol{A}^{-1}$ 再用 (6.59) 式求解 $\boldsymbol{x}$ 初看起来似乎合理. 但是, 由于求 $\boldsymbol{A}^{-1}$ 需要花费的运算量更大, 因此这不是一个有效的方法. 可以证明, 求 $\boldsymbol{A}^{-1}$ 花费的运算量为

$$MD(\boldsymbol{A} \to \boldsymbol{A}^{-1}) = n^3$$

这大约是用高斯消元法求 $\boldsymbol{x}$ 工作量的 3 倍. 即使对 n 个不同的右端 $\boldsymbol{b}$ 用 (6.59) 求解 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, 它也没有用高斯消元法仅做一次从 (6.42) 到 (6.46) 中 $\boldsymbol{A} \to \boldsymbol{U}$ 消元步骤更有效. 由 (6.58) 式, 消元法的运算量 (当 $\boldsymbol{A} \to \boldsymbol{U}$ 已经完成) 仅为 n^2, 而这正好是计算 $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ 乘法的次数. 因此, 利用 $\boldsymbol{A}^{-1}$ 求解没有节省运算量.

$\boldsymbol{A}^{-1}$ 的主要作用是作为检验非奇异线性方程组的解的一个理论工具. 除了一些例外情况, 很少需要明确地计算 $\boldsymbol{A}^{-1}$.

MATLAB 程序　下面的 MATLAB 程序执行部分选主元消元法. 用户提供系数矩阵 $\boldsymbol{A}$ 和右端向量 $\boldsymbol{b}$. 程序计算相应的线性方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ 的解. 它也给出一个矩阵存放部分选主元高斯消元法得到的上三角矩阵的上三角部分以及消元中使用的乘数. 第 3 个输出是一个向量, 记录部分选主元中所作的行置换.

```
function [x,lu,piv] = GEpivot(A,b)
%
% function [x,lu,piv] = GEpivot(A,b)
%
% This program employs the Gaussian elimination method with
% partial pivoting to solve the linear system Ax=b.
%
% Input
% A: coefficient square matrix
% b:  right side column vector
%
% Output
% x:  solution vector
% lu:  a matrix whose upper triangular part is the upper
% triangular matrix resulting from the Gaussian
% elimination with partial pivoting, and whose strictly
% lower triangular part stores the multipliers.  In
% other words, it stores the LU factorization of the
% matrix A with row permutations determined by the
% pivoting vector piv.
% piv:  a pivoting vector recording the row permutations.
% check the order of the matrix and the size of the vector
[m,n] = size(A);
if m ~= n
  error('The matrix is not square.')
end
```

```
m = length(b);
if m ~= n
  error('The matrix and the vector do not match in size.')
end

% initialization of the pivoting vector
piv = (1:n)';

% elimination step
for k = 1:n-1
% Find the maximal element in the pivot column, below the
% pivot position, along with the index of that maximal element.

  [col_max index] = max(abs(A(k:n,k)));
  index = index + k-1;
  if index ~= k
%   Switch rows k and index, in columns k through n.
%   Do similarly for the right-hand side b.

    tempA = A(k,k:n);
    A(k,k:n) = A(index,k:n);
    A(index,k:n) = tempA;

    tempb = b(k);
    b(k) = b(index);
    b(index) = tempb;

    temp = piv(k);
    piv(k) = piv(index);
    piv(index) = temp;
  end

% Form the needed multipliers and store them into the pivot
% column, below the diagonal.
  A(k+1:n,k) = A(k+1:n,k)/A(k,k);

% Carry out the elimination step, first modifying the matrix,
% and then modifying the right-hand side.
  for i = k+1:n
    A(i,k+1:n) = A(i,k+1:n) - A(i,k)*A(k,k+1:n);
  end
  b(k+1:n) = b(k+1:n) - A(k+1:n,k)*b(k);
end

% Solve the upper triangular linear system.
x = zeros(n,1);
```

```
x(n) = b(n)/A(n,n);
for i = n-1:-1:1
  x(i) = (b(i)-A(i,i+1:n)*x(i+1:n))/A(i,i);
end

% Record the LU factorization with row permutation.
lu = A;
```

习题

1. 修改 GEpivot 程序, 去掉选主元部分. 然后, 使用修改的程序用不选主元高斯消元法求解下列线性方程组. 方程组由 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 指定, 然后把方程组写成 (6.42) 的形式.

(a)
$$\boldsymbol{A}=\begin{bmatrix} 2 & 1 & -1 \\ 4 & 0 & -1 \\ -8 & 2 & 2 \end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix} 6 \\ 6 \\ -8 \end{bmatrix}$$

(b)
$$\boldsymbol{A}=\begin{bmatrix} 2 & 1 & -1 & -2 \\ 4 & 4 & 1 & 3 \\ -6 & -1 & 10 & 10 \\ -2 & 1 & 8 & 4 \end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix} 2 \\ 4 \\ -5 \\ 1 \end{bmatrix}$$

(c)
$$\boldsymbol{A}=\begin{bmatrix} 1 & -1 & 2 \\ -1 & 5 & 4 \\ 2 & 4 & 29 \end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix} 1 \\ -3 \\ 15 \end{bmatrix}$$

2. 对于 $n=2,5,10,20$, 用本节中给出的 GEpivot 程序求解 n 阶线性方程组, 其中系数由 6.1 节的 (6.3) 到 (6.4) 给出.
3. 重复习题 2, 但使用下式给出的数组 $\boldsymbol{A}$ 和 $\boldsymbol{b}$

$$a_{ij}=\min\{i,j\},\quad b_i=1,\quad 1\leqslant i,j\leqslant n$$

4. 对于 $n=3$, 假设求解方程组 (6.39) 第 2 步中 $a_{22}^{(2)}$ 和 $a_{32}^{(2)}$ 都是零. 则方程组 (6.40) 变为

$$\begin{cases} a_{11}x_1+a_{12}x_2+a_{13}x_3=b_1 \\ \qquad\qquad\qquad a_{23}^{(2)}x_3=b_2^{(2)} \\ \qquad\qquad\qquad a_{33}^{(2)}x_3=b_3^{(2)} \end{cases}$$

证明这个方程组不是唯一可解的, 而是无解或有无穷多个解. 同样的论证可用于更大型的线性方程组, 这样非奇异线性方程组必有一个非零主元.

5. 对于具有下列系数的方程组重复习题 2:

(a) $$A=\begin{bmatrix}5&7&6&5\\7&10&8&7\\6&8&10&9\\5&7&9&10\end{bmatrix},\quad b=\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix},\quad x=\begin{bmatrix}136\\-82\\-35\\21\end{bmatrix}$$

(b) $$A=\begin{bmatrix}1&\frac{1}{2}&\frac{1}{3}&\frac{1}{4}\\\frac{1}{2}&\frac{1}{3}&\frac{1}{4}&\frac{1}{5}\\\frac{1}{3}&\frac{1}{4}&\frac{1}{5}&\frac{1}{6}\\\frac{1}{4}&\frac{1}{5}&\frac{1}{6}&\frac{1}{7}\end{bmatrix},\quad b=\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix},\quad x=\begin{bmatrix}516\\-5\,700\\13\,620\\-8\,820\end{bmatrix}$$

将你计算的答案与给出的真解作比较. 打印用计算机算术运算得到的全部数字. 注意在计算值中的任何异常情况.

6. 通过手算计算下列矩阵的逆矩阵:

(a) $\begin{bmatrix}3&1&1\\1&3&1\\1&1&3\end{bmatrix}$ (b) $\begin{bmatrix}0&1&2\\1&0&1\\2&1&0\end{bmatrix}$ (c) $\begin{bmatrix}1&2&4\\1&3&9\\1&4&16\end{bmatrix}$

(d) $\begin{bmatrix}-3&1&0&0\\1&-2&1&0\\0&0&-2&1\\0&0&1&-1\end{bmatrix}$ (e) $\begin{bmatrix}1&1&1&1\\1&2&3&4\\1&3&6&10\\1&4&10&20\end{bmatrix}$

7. 在方程组 (6.42) 中, 当 $i-j\geqslant 2$ 时, 规定 $a_{ij}=0$. 写出这个方程组的一般形式. 用高斯消元法求解方程组时, 利用零元已知这一点. 求此情况下 $\boldsymbol{A}\to\boldsymbol{U}$ 的运算量.

6.4 *LU* 分 解

当我们使用矩阵乘法时, 对 6.3 节高斯消元法可以给出另一种表述. 线性方程组的系数矩阵能分解为两个三角形矩阵之积. 本节从这个结论开始, 然后讨论它的应用. 如前面一样, 用 $\boldsymbol{Ax}=\boldsymbol{b}$ 表示所求解的方程组, 其中 $\boldsymbol{A}$ 是 $n\times n$ 系数矩阵. 在 6.3 节的消元步骤 (6.42) 到 (6.45) 中, 线性方程组化为 (6.46) 形式的上三角形方程组 $\boldsymbol{Ux}=\boldsymbol{g}$, 其中

$$\boldsymbol{U}=\begin{bmatrix}u_{11}&u_{12}&\cdots&u_{1n}\\0&u_{22}&\cdots&u_{2n}\\\vdots&\ddots&\ddots&\vdots\\0&\cdots&0&u_{nn}\end{bmatrix}\tag{6.60}$$

并且, $u_{ij}=a_{ij}^{(i)}$. 引进存放 (6.44) 乘数 m_{ij} 的辅助下三角形矩阵 L:

$$\boldsymbol{L}=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ m_{n1} & \cdots & m_{n,n-1} & 1 \end{bmatrix} \tag{6.61}$$

下列定理给出矩阵 $\boldsymbol{L}$ 和矩阵 $\boldsymbol{U}$ 与原始矩阵 $\boldsymbol{A}$ 的关系.

定理 6.4.1　设 $\boldsymbol{A}$ 是一个非奇异矩阵, 并且由 (6.60) 和 (6.61) 定义 $\boldsymbol{L}$ 和 $\boldsymbol{U}$. 如果 $\boldsymbol{U}$ 如 (6.42) 到 (6.45) 那样不选主元得到, 那么

$$\boldsymbol{LU}=\boldsymbol{A} \tag{6.62}$$

这称为 $\boldsymbol{A}$ 的 $\boldsymbol{LU}$ 分解.

我们省略证明, 但习题 3 和习题 4 考虑 $n=3$ 的情况. 同样, 当选主元时有与 (6.62) 类似的结论, 但这里并不考虑它.

例 6.4.2　回顾方程组 (6.48). 则有

$$\boldsymbol{L}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{3}{4} & 1 & 0 & 0 \\ \frac{1}{2} & \frac{6}{7} & 1 & 0 \\ \frac{1}{4} & \frac{5}{7} & \frac{5}{6} & 1 \end{bmatrix},\quad \boldsymbol{U}\begin{bmatrix} 4 & 3 & 2 & 1 \\ 0 & \frac{7}{4} & \frac{3}{2} & \frac{5}{4} \\ 0 & 0 & \frac{12}{7} & \frac{10}{7} \\ 0 & 0 & 0 & \frac{5}{3} \end{bmatrix} \tag{6.63}$$

并且

$$\boldsymbol{LU}=\begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}=\boldsymbol{A}$$

■

$\boldsymbol{LU}$ 分解导出另一种观点的高斯消元法. 由 (6.62), 求解 $\boldsymbol{Ax}=\boldsymbol{b}$ 等价于求解 $\boldsymbol{LUx}=\boldsymbol{b}$. 而这等价于求解两个方程组

$$\boldsymbol{Lg}=\boldsymbol{b},\quad \boldsymbol{Ux}=\boldsymbol{g} \tag{6.64}$$

$\boldsymbol{Ux}=\boldsymbol{g}$ 是上三角形方程组, 它的解基于回代在 (6.47) 中给出. $\boldsymbol{Lg}=\boldsymbol{b}$ 是下三角形方程组

$$\begin{cases} g_1 & =b_1 \\ m_{21}g_1+\quad g_2 & =b_2 \\ \qquad\vdots & \quad\vdots \\ m_{n1}g_1+m_{n2}g_2+\cdots+m_{n,n-1}g_{n-1}+g_n & =b_n \end{cases}$$

由向前替换解此方程组, 得到 $\boldsymbol{Ux}=\boldsymbol{g}$ 中列向量 $\boldsymbol{g}$. 因此, 一旦知道分解 $\boldsymbol{A}=\boldsymbol{LU}$, 那么 $\boldsymbol{Ax}=\boldsymbol{b}$ 的解简化为求解两个比较简单且计算量较少的三角形方程组.

例 6.4.3 再回顾方程组 (6.48), 这里

$$\boldsymbol{b}=\begin{bmatrix}1, & 1, & -1, & -1\end{bmatrix}^{\mathrm{T}}$$

使用 (6.63) 给出的分解 $\boldsymbol{A}=\boldsymbol{LU}$. 则 $\boldsymbol{Lg}=\boldsymbol{b}$ 的解为

$$\boldsymbol{g}=\begin{bmatrix}1, & \frac{1}{4}, & -\frac{12}{7}, & 0\end{bmatrix}^{\mathrm{T}}$$

而 $\boldsymbol{Ux}=\boldsymbol{g}$ 的解为

$$\boldsymbol{x}=\begin{bmatrix}0, & 1, & -1, & 0\end{bmatrix}^{\mathrm{T}}$$

读者应该检查这些结果. ■

6.4.1 高斯消元法的紧凑变形

除了能用 6.3 节的消元步骤 (6.42) 到 (6.45) 来构造 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 外, 也可以直接求出这些矩阵. 运算量与高斯消元法一样, 但这种高斯消元法的*紧凑变形* (compact varicants) 有它的优点.

为了说明怎样直接计算 $\boldsymbol{L}$ 和 $\boldsymbol{U}$, 考虑 $n=3$ 的情况. 把 $\boldsymbol{A}=\boldsymbol{LU}$ 写成

$$\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ m_{21} & 1 & 0\\ m_{31} & m_{32} & 1\end{bmatrix}\begin{bmatrix}u_{11} & u_{12} & u_{13}\\ 0 & u_{22} & u_{23}\\ 0 & 0 & u_{33}\end{bmatrix}$$

把 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 相乘, 并比较乘积的元素与 $\boldsymbol{A}$ 中相应的元素. 比较乘积的第 1 行, 得到

$$a_{11}=u_{11},\quad a_{12}=u_{12},\quad a_{13}=u_{13} \tag{6.65}$$

这就给出 $\boldsymbol{U}$ 的第 1 行. 接下来, 将 $\boldsymbol{L}$ 的第 2 行和 $\boldsymbol{U}$ 相乘, 得到

$$a_{21}=m_{21}u_{11},\quad a_{22}=m_{21}u_{12}+u_{22},\quad a_{23}=m_{21}u_{13}+u_{23} \tag{6.66}$$

从这些关系式可以解出未知的 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的第 2 行元素 m_{21},u_{22} 以及 u_{23}. 最后, $\boldsymbol{L}$ 的第 3 行和 $\boldsymbol{U}$ 相乘得到

$$\begin{aligned}a_{31}&=m_{31}u_{11}\\ a_{32}&=m_{31}u_{12}+m_{32}u_{22}\\ a_{33}&=m_{31}u_{13}+m_{32}u_{23}+u_{33}\end{aligned} \tag{6.67}$$

这些方程得到 m_{31}, m_{32} 以及 u_{33} 的值, 这样就完成 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的构造. 在这个过程中, 为了求 $\boldsymbol{L}$ 必须假设 $u_{11} \neq 0, u_{22} \neq 0$. 为了避免这个假设可用选主元来修改方法, 但我们只考虑相关元素非零的简单情况.

例 6.4.4　设

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix}$$

利用 (6.65) 得到

$$u_{11} = a_{11} = 1, \quad u_{12} = a_{12} = 1, \quad u_{13} = a_{13} = -1$$

由 (6.66) 和 (6.67) 得,

$$\begin{aligned}
m_{21} &= \frac{a_{21}}{u_{11}} = 1 \\
u_{22} &= a_{22} - m_{21}u_{12} = 2 - 1 \times 1 = 1 \\
u_{23} &= a_{23} - m_{21}u_{13} = -2 - 1 \times (-1) = -1 \\
m_{31} &= \frac{a_{31}}{u_{11}} = -2 \\
m_{32} &= (a_{32} - m_{31}u_{12})/u_{22} = (1 - (-2) \times 1)/1 = 3 \\
u_{33} &= a_{33} - m_{31}u_{13} - m_{32}u_{23} = 1 - (-2) \times (-1) - 3 \times (-1) = 2
\end{aligned}$$

因此,

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 2 \end{bmatrix}$$ ■

对任何 $n \geqslant 2$ 可推广上面的过程 (6.65) 到 (6.67), 所得到的方法称为**杜立德方法** (Doolittle's method). 也可以将它修改来对应于 6.3 节中讨论的选主元方法. 这里不给出杜立德方法的详细定义, 但它比高斯消元法具有计算上的优点.

作为计算 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的两种方法, 高斯消元法和杜立德方法数学上是等价的并且有相同的运算量. 但是使用杜立德方法可以在少量增加工作量的前提下, 大大减少舍入误差. 对于较大的 n 值, 如 (6.67) 显示的那样, 计算 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的公式包含许多乘积之和. 如果假设那些数全都以单精度存放, 那么可以使用限定的双精度大大减少单精度舍入误差次数. 可用双精度计算所有的乘积与求和, 然后舍入到单精度. 因此, 单精度数乘积之和计算仅包含一个而不是许多个单精度舍入误差. 这种舍入误差次数从正规高斯消元法大约为 $\frac{2}{3}n^3$ 减到对杜立德方法的 n^2 倍数, 对较大的 n 次数减少更多. 在第 2 章末已讨论过使用限定的双精度算术运算. 求解线性方程组

较好的程序是使用杜立德方法或与它密切相关的方法, 并用上面描述的限定的双精度算术运算.

再来讨论矩阵 $\boldsymbol{L}$ 的存储. 回到 6.3 节的消元步骤 (6.42) 到 (6.45), 我们看到乘数 m_{ik} 存放在消元步骤中化为零的机器数组 $\boldsymbol{A}$ 的存储位置. 因此, 在机器中, 对于 $i > k$, $\boldsymbol{A}(i,k)$ 存放 m_{ik}. 如前面讨论的那样, 对于 $j \geqslant i$, $\boldsymbol{A}(i,j)$ 存放 u_{ij}. $\boldsymbol{L}$ 的对角元素都是 1, 所以它们不需要存放. 因此, 对 $\boldsymbol{Ax} = \boldsymbol{b}$ 应用高斯消元法或杜立德方法之后, 如 6.3 节中 `GEpivot` 程序那样, $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的元素将取代计算机中 $\boldsymbol{A}$ 的原始元素.

6.4.2 三对角方程组

当系数矩阵 $\boldsymbol{A}$ 具有特殊形式时, 常可简化高斯消元法. 我们以三对角线性方程组为例进行讨论.

方程组 $\boldsymbol{Ax} = \boldsymbol{f}$ 称为*三对角的*, 如果它的系数矩阵具有形式

$$\boldsymbol{A} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & \cdots & 0 & a_n & b_n \end{bmatrix} \tag{6.68}$$

这个矩阵被称为*三对角矩阵*. *三对角方程组*通常出现在求解数值分析的许多问题中. 在第 5 章, 样条函数插值导出 (4.64) 中的三对角方程组. 二阶常微分方程两点边值问题的有限差分解也导出三对角方程组, 见 8.8 节.

对 $\boldsymbol{Ax} = \boldsymbol{f}$ 用高斯消元法时, 大部分乘数 $m_{ik} = 0$ 并且 $\boldsymbol{U}$ 的大部分元素也是零. 可以证明, 如果不需要选主元, 则 $\boldsymbol{LU}$ 分解具有下面的一般形式:

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \alpha_2 & 1 & 0 & \cdots & 0 \\ 0 & \alpha_3 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n & 1 \end{bmatrix} \cdots \begin{bmatrix} \beta_1 & c_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & c_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \vdots & 0 & \beta_{n-1} & c_{n-1} \\ 0 & \cdots & \cdots & 0 & \beta_n \end{bmatrix} \tag{6.69}$$

用 $\boldsymbol{L}$ 的每一行逐次乘 $\boldsymbol{U}$ 的各列, 所得结果等于 $\boldsymbol{A}$ 相应的元素. 则得到

$$\begin{aligned} &\beta_1 = b_1 && : \ LU \text{ 的第 } 1 \text{ 行} \\ &\alpha_2\beta_1 = a_2, \quad \alpha_2 c_1 + \beta_2 = b_2 && : \ LU \text{ 的第 } 2 \text{ 行} \\ &\alpha_j\beta_{j-1} = a_j, \quad \alpha_j c_{j-1} + \beta_j = b_j && : \ LU \text{ 的第 } j \text{ 行}, j = 3, \cdots, n \end{aligned} \tag{6.70}$$

读者应该对 $\boldsymbol{A}$ 的前几行核对这些等式.

从方程 (6.70) 容易算出元素 $\{\alpha_i\}$ 和 $\{\beta_i\}$.

$$\begin{aligned}&\beta_1 = b_1\\&\alpha_j = a_j/\beta_{j-1}, \quad \beta_j = b_j - \alpha_j c_{j-1}, \quad j = 2, \cdots, n\end{aligned} \tag{6.71}$$

这样, 方程组 $\boldsymbol{Ax} = \boldsymbol{f}$ 转化为一对三角形方程组

$$\boldsymbol{Lg} = \boldsymbol{f}, \quad \boldsymbol{Ux} = \boldsymbol{g}$$

对 $\boldsymbol{Lg} = \boldsymbol{f}$ 向前替换, 得到

$$g_1 = f_1, \quad g_j = f_j - \alpha_j g_{j-1}, \quad j = 2, 3, \cdots, n \tag{6.72}$$

对 $\boldsymbol{Ux} = \boldsymbol{g}$ 回代, 得到

$$x_n = \frac{g_n}{\beta_n}, \quad x_j = \frac{g_j - c_j x_{j+1}}{\beta_j}, \quad j = n-1, \cdots, 1 \tag{6.73}$$

从 (6.71) 到 (6.73) 的整个计算过程非常迅速, 只有 $5n-4$ 次乘除法的运算工作量. 这个结论的证明留作习题 11. 为存放矩阵 $\boldsymbol{A}$, 仅需要 3 个一维数组存放 $\boldsymbol{A}$ 的三条对角线. 这意味可以很容易地求解大型的三对角方程组. 在一些应用中, 例如, 求解微分方程边值问题中, 阶数超过 $n = 10\ 000$ 的方程组并不罕见.

例 6.4.5　设

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 & 0 & 0 & 0\\ 1 & 2 & 1 & 0 & 0\\ 0 & 1 & 2 & 1 & 0\\ 0 & 0 & 1 & 2 & 1\\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} \tag{6.74}$$

按 (6.68) 中记号, 对所有 j

$$a_j = 1, \quad b_j = 2, \quad c_j = 1,$$

由 (6.71), $\beta_1 = 2$,

$$\alpha_j = \frac{1}{\beta_{j-1}}, \quad \beta_j = 2 - \alpha_j, \quad j = 2, 3, 4, 5$$

得到

$$\boldsymbol{A} = \boldsymbol{LU} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0\\ \frac{1}{2} & 1 & 0 & 0 & 0\\ 0 & \frac{2}{3} & 1 & 0 & 0\\ 0 & 0 & \frac{3}{4} & 1 & 0\\ 0 & 0 & 0 & \frac{4}{5} & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & 0 & 0\\ 0 & \frac{3}{2} & 1 & 0 & 0\\ 0 & 0 & \frac{4}{3} & 1 & 0\\ 0 & 0 & 0 & \frac{5}{4} & 1\\ 0 & 0 & 0 & 0 & \frac{6}{5} \end{bmatrix}$$ ■

假设过程 (6.69) 到 (6.73) 不必选主元. 可以证明, 如果满足下列条件

$$
\begin{aligned}
&|b_1| > |c_1| > 0 \\
&|b_j| \geqslant |a_j| + |c_j| \ , \quad a_j, c_j \neq 0, \quad j = 2, 3, \cdots, n-1 \\
&|b_n| > |a_n| > 0
\end{aligned}
\tag{6.75}
$$

则可不必选主元. 应用中出现的许多重要情况满足上述这些条件. 习题 12 考虑三次样条插值情况.

稀疏 (sparse) **线性方程组**是指系数矩阵中非零元素所占百分比非常小的方程组. 三对角线性方程组就是稀疏方程组的一个例子. 类似于三对角方程组, 也可以把高斯消元法有效地推广到求解其他稀疏形式的线性方程组; 但这里不加以考虑.

MATLAB 程序 下列 MATLAB 程序利用方法 (6.69) 到 (6.73) 解三对角方程组. 用户以向量 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ 给出系数 $\{a_j, b_j, c_j\}$, 并且在 $\boldsymbol{F}$ 中提供右端常数. 程序运行后, 向量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 包含 (6.71) 的常数 $\{\alpha_j, \beta_j\}$, 而 $\boldsymbol{F}$ 包含解 $\boldsymbol{x}$. 在许多应用中, 要求解具有许多右端 $\boldsymbol{f}$ 的三对角方程组. 为了允许这种选择, 变量 `iflag` 表示 ***LU*** 分解是否已经完成计算并放到数组 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ 中. 附加的信息见程序中注解语句. 8.8 节求解常微分方程两点边值问题的有限差分方程组将使用这个程序.

```
function [x, ier, alpha, beta] = tridiag(a,b,c,f,n,iflag)
%
% function [x, ier, alpha, beta] = tridiag(a,b,c,f,n,iflag)
%
% Solve a tridiagonal linear system M*x=f
%
% INPUT:
% The order of the linear system is given as n.
% The subdiagonal, diagonal, and superdiagonal of M are given
% by the arrays a,b,c, respectively.  More precisely,
% M(i,i-1) = a(i), i=2,...,n
% M(i,i) = b(i), i=1,...,n
% M(i,i+1) = c(i), i=1,...,n-1
% iflag=0 means that the original matrix M is given as
% specified above.
% iflag=1 means that the LU factorization of M is already known
% and is stored in a,b,c.  This will have been accomplished by
% a previous call to this routine.  In that case, the vectors
% alpha and beta should have been substituted for a and b
% in the calling sequence.
%
% OUTPUT:
% Upon exit, the LU factorization of M is already known and
% is stored in alpha,beta,c.  The solution x is given as well.
```

```
% ier=0 means the program was completed satisfactorily.
% ier=1 means that a zero pivot element was encountered and the
% solution process was abandoned.

a(1) = 0;
if iflag == 0

% Compute LU factorization of matrix M.
  for j = 2:n
    if b(j-1) == 0
      ier = 1;
      return
    end
    a(j) = a(j)/b(j-1);
    b(j) = b(j) - a(j)*c(j-1);
  end
  if b(n) == 0
    ier = 1;
    return
  end
end

% Compute solution x to M*x = f.
% Do forward substitution to solve lower triangular system.
for j = 2:n
  f(j) = f(j) - a(j)*f(j-1);
end

% Do backward substitution to solve upper triangular system.
f(n) = f(n)/b(n);
for j = n-1:-1:1
  f(j) = (f(j) - c(j)*f(j+1))/b(j);
end

% Set output variables.
ier = 0;
x = f;
alpha = a; beta = b;
```

6.4.3 解线性方程组的 MATLAB 内置函数

对于具有非奇异方阵 $\boldsymbol{A}$ 的线性方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, 通过左除运算:

```
x = A \ b
```

可以得到解 $\boldsymbol{x}$. 当 $\boldsymbol{A}$ 是三角形矩阵, 或者是通过行交换得到的三角形矩阵时, 通过

代换计算得出方程组的解. 如果 $\boldsymbol{A}$ 是对称正定的, 则计算 $\boldsymbol{A}$ 的 Cholesky 分解 (见习题 6), 再通过代入求解得到的两个三角形方程组. 对一般非奇异方阵 $\boldsymbol{A}$, 根据 $\boldsymbol{A}$ 的 $\boldsymbol{LU}$ 分解, 再作代入来求解. 当发觉矩阵接近于奇异时, 会对可能不准确的结果发出警告信息.

MATLAB 用 `lu` 函数进行 $\boldsymbol{LU}$ 分解 (用部分选主元). 对于方阵 $\boldsymbol{A}$, 调用

```
[L, U, P] = lu(A)
```

提供具有单位对角元的下三角矩阵 $\boldsymbol{L}$、上三角矩阵 $\boldsymbol{U}$ 以及记录分解中所用行交换的置换矩阵 $\boldsymbol{P}$. 输出矩阵满足关系 $\boldsymbol{PA} = \boldsymbol{LU}$. 调用 `lu` 函数的另一种形式是

```
[L, U] = lu(A)
```

则 $\boldsymbol{LU} = \boldsymbol{A}$, 并且 $\boldsymbol{L}$ 是具有单位对角元的下三角矩阵的一个置换, $\boldsymbol{U}$ 是上三角矩阵.

当需要计算非奇异矩阵的逆时, 可调用 `inv` 函数求逆. 如 6.3.3 节讨论的那样, 用逆阵求解给定的线性方程组是不可取的. 通常, 计算

```
x = A \ b
```

比计算

```
x = inv(A) * b
```

快 2 到 3 倍. 鼓励读者用这些函数和其他关于线性方程组求解的 MATLAB 内置函数进行实验.

习题

1. 回顾 6.3 节方程组 (6.35) 的解. 使用那种计算, 给出那个方程组的系数矩阵的 $\boldsymbol{LU}$ 分解.
2. 计算 6.3 节习题 1 中矩阵的 $\boldsymbol{LU}$ 分解.
3. 考虑当 $n = 3$ 时定理 6.4.1 的证明. 回顾对三个方程构成的方程组的高斯消元法推导过程 (6.39) 到 (6.41). 构成乘积

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{bmatrix}$$

用 m_{ij} 和 $a_{ij}^{(i)}$ 的定义简化结果.

4. 回顾 6.2 节习题 30 中初等矩阵的定义. 引进

$$\boldsymbol{E}_1 = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{bmatrix}, \quad \boldsymbol{E}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -m_{32} & 1 \end{bmatrix}$$

并定义 $\boldsymbol{A}_2=\boldsymbol{E}_1\boldsymbol{A}$, $\boldsymbol{A}_3=\boldsymbol{E}_2\boldsymbol{A}_2$. 证明由高斯消元法得到的上三角矩阵 $\boldsymbol{A}_3=\boldsymbol{U}$. 因此, $\boldsymbol{E}_2\boldsymbol{E}_1\boldsymbol{A}=\boldsymbol{U}$. 利用这个结论证明 $\boldsymbol{A}=\boldsymbol{LU}$, 其中 $\boldsymbol{L}$ 由 $\boldsymbol{E}_2$ 和 $\boldsymbol{E}_1$ 导出. 怎样将这证明推广到 n 阶矩阵 $\boldsymbol{A}$?

5. 如果仅要求 $\boldsymbol{L}$ 是下三角矩阵和 $\boldsymbol{U}$ 是上三角矩阵, 则 $\boldsymbol{A}$ 的 $\boldsymbol{LU}$ 分解不唯一.

(a) 已知 $\boldsymbol{A}$ 的 $\boldsymbol{LU}$ 分解, 对某个非奇异对角矩阵 $\boldsymbol{D}$, 定义

$$\boldsymbol{L}_1=\boldsymbol{LD},\quad \boldsymbol{U}_1=\boldsymbol{D}^{-1}\boldsymbol{U}$$

证明 $\boldsymbol{L}_1\boldsymbol{U}_1$ 是 $\boldsymbol{A}$ 的另一个这种分解.

(b) 设 $\boldsymbol{L}_1\boldsymbol{U}_1=\boldsymbol{L}_2\boldsymbol{U}_2=\boldsymbol{A}$, 其中 $\boldsymbol{L}_1,\boldsymbol{L}_2$ 是下三角矩阵, $\boldsymbol{U}_1,\boldsymbol{U}_2$ 是上三角矩阵. 同样, 设 $\boldsymbol{A}$ 是非奇异矩阵. 证明对某个对角矩阵 $\boldsymbol{D}$, $\boldsymbol{L}_1=\boldsymbol{L}_2\boldsymbol{D}$, $\boldsymbol{U}_1=\boldsymbol{D}^{-1}\boldsymbol{U}_2$.

提示: 推导 $\boldsymbol{L}_2^{-1}\boldsymbol{L}_1=\boldsymbol{U}_2\boldsymbol{U}_1^{-1}$. 证明这些矩阵一定是对角形矩阵. 为了使问题更简单, 先设 $\boldsymbol{A}$ 的阶数 $n=2$. 并且回顾 6.2 节习题 3.

6. 当 $\boldsymbol{A}$ 是对称正定矩阵时, 可以把它分解为

$$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$$

其中 $\boldsymbol{L}$ 是某个下三角矩阵, 其对角元素非零, 通常取正数. 这称为楚列斯基 (Cholesky) 分解. 回顾: 如果对任何向量 $\boldsymbol{x}\neq\boldsymbol{0},\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}>0$, 则称对称矩阵 $\boldsymbol{A}$ 是正定的.

(a) 对于

$$\boldsymbol{A}=\begin{bmatrix}1 & -1\\ -1 & 5\end{bmatrix}$$

设

$$\boldsymbol{L}=\begin{bmatrix}l_{11} & 0\\ l_{21} & l_{22}\end{bmatrix}$$

求 $\boldsymbol{L}$, 使得 $\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}=\boldsymbol{A}$.

(b) 对于

$$\boldsymbol{A}=\begin{bmatrix}2.25 & -3 & 4.5\\ -3 & 5 & -10\\ 4.5 & -10 & 34\end{bmatrix}$$

重复这过程, 求下三角矩阵 $\boldsymbol{L}$, 使得 $\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}=\boldsymbol{A}$.

7. 编写求解线性方程组 (6.64) 的 MATLAB 代码. 假设 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 按照本节前面所讨论的三角形方程组的约定存放.

8. 对于 (6.74) 中的矩阵 $\boldsymbol{A}$, 以及

$$\boldsymbol{f}=[1,0,0,0,0]^{\mathrm{T}}$$

求解线性方程组.

9. 求解三对角方程组

$$\begin{cases}2x_1+x_2=3\\ -x_1+2x_2+x_3=2\\ -x_2+2x_3+x_4=2\\ -x_3+2x_4+x_5=2\\ -x_4+2x_5=1\end{cases}$$

10. 求解三对角方程组 $\boldsymbol{Ax}=\boldsymbol{f}$, 其中对所有 i

$$\boldsymbol{A}_{ii}=4,\quad \boldsymbol{A}_{i,i-1}=\boldsymbol{A}_{i,i+1}=1$$

设方程组阶数 $n=100$, 并设

$$\boldsymbol{f}=[1,1,\cdots,1]^{\mathrm{T}}$$

11. 设 $\boldsymbol{Ax}=\boldsymbol{f}$ 是一个 n 阶三对角方程组.

(a) 给出利用 (6.71) 形成 $\boldsymbol{A}=\boldsymbol{LU}$ 的运算量.

(b) 假设 $\boldsymbol{LU}$ 已知, 给出利用 (6.72) 到 (6.73) 求解 $\boldsymbol{Ax}=\boldsymbol{f}$ 的运算量.

12. 回顾三次样条插值的三对角方程组 (4.64) 到 (4.65). 证明这方程组满足条件 (6.75).

6.5 解线性方程组中的误差

在计算机或计算器上, 高斯消元法的整个算术运算都包含舍入误差, 这些误差导致所求解的线性方程组计算解中的误差. 本节中, 我们给出估计和校正计算解中误差的一种方法. 然后, 我们对于诸如舍入误差那样小的变化检验解的敏感性.

设 $\boldsymbol{Ax}=\boldsymbol{b}$ 表示要求解的线性方程组, 并设 $\hat{\boldsymbol{x}}$ 表示计算解. 定义

$$\boldsymbol{r}=\boldsymbol{b}-\boldsymbol{A}\hat{\boldsymbol{x}} \tag{6.76}$$

$\boldsymbol{r}$ 称为 $\boldsymbol{A}\hat{\boldsymbol{x}}$ 逼近 $\boldsymbol{b}$ 的*残差*. 如果 $\hat{\boldsymbol{x}}$ 是精确解, 则 $\boldsymbol{r}$ 等于 0. 为把 $\boldsymbol{r}$ 与解的误差联系起来, 将 $\boldsymbol{b}=\boldsymbol{Ax}$ 代入 (6.76) 中, 得到

$$\boldsymbol{r}=\boldsymbol{Ax}-\boldsymbol{A}\hat{\boldsymbol{x}}=\boldsymbol{A}(\boldsymbol{x}-\hat{\boldsymbol{x}})$$

记 $\hat{\boldsymbol{x}}$ 的误差 $\boldsymbol{e}=\boldsymbol{x}-\hat{\boldsymbol{x}}$. 则误差 $\boldsymbol{e}$ 满足线性方程组

$$\boldsymbol{Ae}=\boldsymbol{r} \tag{6.77}$$

它与原始方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 具有同样的系数矩阵 $\boldsymbol{A}$.

为了得到 $\boldsymbol{e}$, 求解 (6.77). 在求解原方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的过程中, 矩阵 $\boldsymbol{L}$ 和矩阵 $\boldsymbol{U}$ 以及一切可能的行交换消息均被存储, 参见 6.3 节中程序 `GEpivot`. 利用这些信息求解 $\boldsymbol{Ae}=\boldsymbol{r}$ 的计算成本相对较小. 回顾 (6.58) 式, 我们知道由 (6.77) 式求 $\boldsymbol{e}$ 的运算量是

$$MD(\boldsymbol{r}\to\boldsymbol{e})=n^2$$

此外, 必须由 (6.76) 式计算残差 $\boldsymbol{r}$, 其运算量为,

$$MD(\boldsymbol{r})=n^2$$

因此, 由 $\hat{x}$ 计算 e 的运算量仅为 $2n^2$, 这个运算量与求解原方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 约为 $\frac{1}{3}n^3$ 的运算量比较相对较小.

实际上实施计算误差 e 的过程中, 有两种可能引起困难的来源. 第一种困难的来源是 (6.76) 中的残差计算会出现有效数字损失的误差. 一般地, 几乎相等的量 b_i 和 $\sum_{j=1}^{n} a_{ij}\hat{x}_j$ 相减使每个分量

$$r_i = b_i - \sum_{j=1}^{n} a_{ij}\hat{x}_j \tag{6.78}$$

非常接近于零. 为了避免得到非常不准确的 r_i 值, 应该采用更高精度的算术运算计算 (6.78) 式. 如果采用单精度算术运算求解 $\boldsymbol{Ax}=\boldsymbol{b}$, 则应该使用双精度的算术运算计算 $\boldsymbol{r}$, 然后再舍入到单精度. 如果用双精度的算术运算求解方程组, 则 $\boldsymbol{r}$ 应该用某种形式的扩展精度算术运算进行计算. IEEE 浮点算术运算标准包含了这种扩展算术运算.

计算 e 的第二种困难的来源是求解 $\boldsymbol{Ae}=\boldsymbol{r}$ 将包含与导致计算 $\hat{x}$ 值的误差一样的舍入误差. 因此, 我们不会得到 e, 而仅仅是得到 e 的一个近似值 $\hat{e}$. $\hat{e}$ 是否很接近 e 依赖于矩阵的敏感性或条件, 这个话题将在本节后面考虑. 作为一般规则, 除了非常病态的线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$, 近似值 $\hat{e}$ 相对于 e 至少有 1 位或多位精确数字.

例 6.5.1 回顾 6.3 节例 3.3.3 中的方程组

$$\begin{cases} 0.729x_1+0.81x_2+0.9x_3=0.686\,7 \\ x_1+\quad x_2+\quad x_3=0.833\,8 \\ 1.331x_1+1.21x_2+1.1x_3=1.000 \end{cases} \tag{6.79}$$

如前, 使用具有舍入功能的 4 位十进制机器. 该方程组恰当地舍入到 4 位有效数字的正确解是

$$x_1=0.224\,5,\quad x_2=0.281\,4,\quad x_3=0.327\,9 \tag{6.80}$$

为了说明误差 e 的估计, 我们考虑用不选主元的高斯消元法求解方程组. 这就导致前面 (6.53) 中给出的解答

$$\hat{x}_1=0.225\,1,\quad \hat{x}_2=0.279\,0,\quad \hat{x}_3=0.329\,5 \tag{6.81}$$

使用 8 位浮点十进制算术运算计算残差 $\boldsymbol{r}$, 并将残差分量舍入到 4 位有效数字, 得到

$$\boldsymbol{r}=[0.000\,062\,10, 0.000\,200\,0, 0.000\,351\,9]^{\mathrm{T}}$$

用这个新的右端向量 $\boldsymbol{r}$ 求解线性方程组 (6.79) 得到近似值

$$\hat{e} = [-0.000\,447\,1, 0.002\,150, -0.001\,504]^{\mathrm{T}}$$

将 $\hat{e}$ 与从 (6.80) 和 (6.81) 得到的真误差

$$\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}} = [-0.000\,7, 0.002\,4, -0.001\,6]^{\mathrm{T}}$$

比较. 可以看出这个 $\hat{e}$ 给出计算解 $\hat{x}$ 的误差 e 大小的一个相当好的办法. ■

6.5.1 残差校正方法

进一步利用这个误差估计方法可以定义一种改进计算值 x 的迭代方法. 记初值 $x^{(0)} = \hat{x}$, 它一般由高斯消元法得到. 定义

$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)} \tag{6.82}$$

则如同 (6.77) 那样, 有

$$\boldsymbol{A}\boldsymbol{e}^{(0)} = \boldsymbol{r}^{(0)}, \quad \boldsymbol{e}^{(0)} = \boldsymbol{x} - \boldsymbol{x}^{(0)}$$

利用高斯消元法求解, 得到一个近似值 $\hat{e}^{(0)} \approx \boldsymbol{e}^{(0)}$, 利用它, 我们定义一个改进的近似值

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \hat{e}^{(0)} \tag{6.83}$$

现在重复整个过程, 计算

$$\boldsymbol{r}^{(1)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(1)}$$

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} + \hat{e}^{(1)}$$

其中 $\hat{e}^{(1)}$ 是

$$\boldsymbol{A}\boldsymbol{e}^{(1)} = \boldsymbol{r}^{(1)}, \quad \boldsymbol{e}^{(1)} = \boldsymbol{x} - \boldsymbol{x}^{(1)} \tag{6.84}$$

的一个近似解. 对 $k \geqslant 0$ 继续这个过程直到 $\hat{e}^{(k)}$ 不再进一步减少, 这里 $k \geqslant 0$.

这个方法通常给出一种手段, 可以得到相对于算术运算中所用的位数尽可能精确的答案. 这个过程称为*残差校正方法*或*迭代改进*.

例 6.5.2 使用具有舍入功能的 4 位浮点十进制算术运算计算机, 并采用选主元的高斯消元法. 求解的方程组是

$$\begin{cases} x_1 + 0.5x_2 + 0.333\,3x_3 = 1 \\ 0.5x_1 + 0.333\,3x_2 + 0.25x_3 = 0 \\ 0.333\,3x_1 + 0.25x_2 + 0.2x_3 = 0 \end{cases} \tag{6.85}$$

则

$$
\begin{aligned}
\boldsymbol{x}^{(0)} &= [8.968, -35.77, 29.77]^{\mathrm{T}} \\
\boldsymbol{r}^{(0)} &= [-0.005\,341, -0.004\,359, -0.000\,534\,4]^{\mathrm{T}} \\
\hat{\boldsymbol{e}}^{(0)} &= [0.092\,16, -0.544\,2, 0.523\,9]^{\mathrm{T}} \\
\boldsymbol{x}^{(1)} &= [9.060, -36.31, 30.29]^{\mathrm{T}} \\
\boldsymbol{r}^{(1)} &= [-0.000\,657\,0, -0.000\,377\,0, -0.000\,198\,0]^{\mathrm{T}} \\
\hat{\boldsymbol{e}}^{(2)} &= [0.001\,707, -0.013\,00, 0.012\,41]^{\mathrm{T}} \\
\boldsymbol{x}^{(2)} &= [9.062, -36.32, 30.30]^{\mathrm{T}}
\end{aligned}
$$

迭代 $\boldsymbol{x}^{(2)}$ 与方程组 (6.85) 的真解舍入以后一致. 这说明残差校正方法非常有用. ■

6.5.2 解线性方程组中的稳定性

上面已经说明, 高斯消元法的舍入误差会导致 $\boldsymbol{Ax}=\boldsymbol{b}$ 计算解的误差. 现在我们想要检查解 $\boldsymbol{x}$ 对于这些误差的敏感程度. 具有许多算术运算的高斯消元法的复杂性使得直接分析这些舍入误差的影响很难. 这里, 我们仅考虑线性方程组右端 $\boldsymbol{b}$ 作出一个小的变化时对方程组解 $\boldsymbol{x}$ 的影响.

设 $\hat{\boldsymbol{b}}$ 是 $\boldsymbol{b}$ 的一个近似值, 考虑

$$\boldsymbol{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$$

的解 $\hat{\boldsymbol{x}}$. 为了检查 $\boldsymbol{x}$ 对 $\boldsymbol{b}$ 的微小变化的敏感性, 我们要将 $\boldsymbol{x}-\hat{\boldsymbol{x}}$ 的大小与 $\boldsymbol{b}-\hat{\boldsymbol{b}}$ 的大小作比较.

例 6.5.3 线性方程组

$$\begin{cases} 5x_1+\ \ 7x_2=0.7 \\ 7x_1+10x_2=1 \end{cases} \tag{6.86}$$

有解

$$x_1=0, \quad x_2=0.1$$

扰动方程组

$$\begin{cases} 5\hat{x}_1+\ \ 7\hat{x}_2=0.69 \\ 7\hat{x}_1+10\hat{x}_2=1.01 \end{cases}$$

有解

$$\hat{x}_1=-0.17, \quad \hat{x}_2=0.22$$

(6.86) 右端一个相对较小的变化导致方程组的解相对较大的改变. ■

为了更精确地表达 $\boldsymbol{x}-\hat{\boldsymbol{x}}$ 与 $\boldsymbol{b}-\hat{\boldsymbol{b}}$ 之间的关系, 我们引进下列关于向量和矩阵大小的度量. 定义

$$\begin{aligned} \|\boldsymbol{z}\| &= \max_{1\leqslant i\leqslant n} |z_i|, \qquad & \boldsymbol{z} \text{ 的阶 } = n\times 1 \\ \|\boldsymbol{A}\| &= \max_{1\leqslant i\leqslant n} \sum_{j=1}^{n} |a_{ij}|, \quad & \boldsymbol{A} \text{ 的阶 } = n\times n \end{aligned} \tag{6.87}$$

上述这些量分别称为**向量和矩阵的范数**. 可以证明它们满足下列不等式

$$\begin{aligned} &\textbf{P1.} \quad \|\boldsymbol{y}+\boldsymbol{z}\| \leqslant \|\boldsymbol{y}\| + \|\boldsymbol{z}\| \\ &\textbf{P2.} \quad \|\boldsymbol{A}+\boldsymbol{B}\| \leqslant \|\boldsymbol{A}\| + \|\boldsymbol{B}\| \\ &\textbf{P3.} \quad \|\boldsymbol{AB}\| \leqslant \|\boldsymbol{A}\|\|\boldsymbol{B}\| \\ &\textbf{P4.} \quad \|\boldsymbol{Az}\| \leqslant \|\boldsymbol{A}\|\|\boldsymbol{z}\| \end{aligned} \tag{6.88}$$

检验误差 $\boldsymbol{x}-\hat{\boldsymbol{x}}$, 我们考虑 $\hat{\boldsymbol{x}}$ 的分量中误差的最大值

$$\|\boldsymbol{x}-\hat{\boldsymbol{x}}\| = \max_{1\leqslant i\leqslant n} |x_i - \hat{x}_i|$$

有许多种不同的向量和矩阵范数可以使用 [见 Atkinson(1989, §7.3)], 但上面 (6.87) 式定义的范数对我们的需要来说足够了.

定理 6.5.4 设 $\boldsymbol{A}$ 非奇异. 则 $\boldsymbol{Ax}=\boldsymbol{b}$ 和 $\boldsymbol{A}\hat{\boldsymbol{x}}=\hat{\boldsymbol{b}}$ 的解满足

$$\frac{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \leqslant \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\frac{\|\boldsymbol{b}-\hat{\boldsymbol{b}}\|}{\|\boldsymbol{b}\|} \tag{6.89}$$

对于适当选择的 $\boldsymbol{b}$ 和 $\hat{\boldsymbol{b}}$, 上面的不等式能成为等式.

证明 从 $\boldsymbol{Ax}=\boldsymbol{b}$ 中减去 $\boldsymbol{A}\hat{\boldsymbol{x}}=\hat{\boldsymbol{b}}$ 得到,

$$\begin{aligned} \boldsymbol{A}(\boldsymbol{x}-\hat{\boldsymbol{x}}) &= \boldsymbol{b}-\hat{\boldsymbol{b}} \\ \boldsymbol{x}-\hat{\boldsymbol{x}} &= \boldsymbol{A}^{-1}(\boldsymbol{b}-\hat{\boldsymbol{b}}) \end{aligned}$$

利用 (6.88) 式的 (P4) 得到

$$\|\boldsymbol{x}-\hat{\boldsymbol{x}}\| = \left\|\boldsymbol{A}^{-1}(\boldsymbol{b}-\hat{\boldsymbol{b}})\right\| \leqslant \|\boldsymbol{A}^{-1}\| \left\|\boldsymbol{b}-\hat{\boldsymbol{b}}\right\|$$

上式两边除以 $\|\boldsymbol{x}\|$, 得到

$$\frac{\|x-\hat{x}\|}{\|x\|} \leqslant \frac{\|\boldsymbol{A}^{-1}\|\|\boldsymbol{b}-\hat{\boldsymbol{b}}\|}{\|\boldsymbol{x}\|} = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\frac{\|\boldsymbol{b}-\hat{\boldsymbol{b}}\|}{\|\boldsymbol{A}\|\|\boldsymbol{x}\|} \tag{6.90}$$

再用 (P4), 得到

$$\|\boldsymbol{b}\| = \|\boldsymbol{Ax}\| \leqslant \|\boldsymbol{A}\|\,\|\boldsymbol{x}\|$$

把它用于 (6.90) 的右端得到 (6.89). 我们略去定理最后关于等式部分的证明. ■

结论 (6.89) 把 $\hat{\boldsymbol{x}}$ 的相对误差与 $\hat{\boldsymbol{b}}$ 的相对误差作比较. 它说明如果 $\|\boldsymbol{A}\|\,\|\boldsymbol{A}^{-1}\|$ 很大, 则 $\hat{\boldsymbol{x}}$ 的相对误差可能比 $\hat{\boldsymbol{b}}$ 的相对误差大很多, 尽管并不是对所有的 $\boldsymbol{b}$ 和 $\hat{\boldsymbol{b}}$, $\hat{\boldsymbol{x}}$ 的相对误差都比 $\hat{\boldsymbol{b}}$ 的相对误差大很多, 但这种情况确实会出现在某些 $\boldsymbol{b}$ 和 $\hat{\boldsymbol{b}}$ 上. 数

$$\operatorname{cond}(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\| \tag{6.91}$$

称为*条件数*. 当它很大时, $\boldsymbol{Ax}=\boldsymbol{b}$ 的解对于 $\boldsymbol{b}$ 相对小的变化很可能会非常敏感. 换言之, 在前面 (6.76) 残差的记号中, 一个相对小的残差很可能导致 $\hat{\boldsymbol{x}}$ 与 $\boldsymbol{x}$ 对比而言相对较大的误差. 这样的线性方程组称为*病态的*, 而具有小条件数的线性方程组称为*良态的*.

当 $\boldsymbol{A}$ 变化而 $\boldsymbol{b}$ 不变时, 我们有类似的结果. 特别地, 比较 $\boldsymbol{Ax}=\boldsymbol{b}$ 与 $\widehat{\boldsymbol{A}}\hat{\boldsymbol{x}}=\boldsymbol{b}$ 的解, 倘若

$$\left\|\boldsymbol{A}-\widehat{\boldsymbol{A}}\right\| < 1/\|\boldsymbol{A}^{-1}\|$$

则我们可以证明

$$\frac{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \leqslant \frac{\operatorname{cond}(\boldsymbol{A})}{1-\operatorname{cond}(\boldsymbol{A})\frac{\left\|\boldsymbol{A}-\widehat{\boldsymbol{A}}\right\|}{\|\boldsymbol{A}\|}} \cdot \frac{\left\|\boldsymbol{A}-\widehat{\boldsymbol{A}}\right\|}{\|\boldsymbol{A}\|}$$

另外, 如果 $\operatorname{cond}(\boldsymbol{A})$ 较大, 则 $\boldsymbol{A}$ 较小的变化可能导致 $\boldsymbol{x}$ 较大的变化. 我们可以利用 MATLAB 的 `cond` 函数来计算条件数. 调用

```
condA = cond(A, inf)
```

给出矩阵 $\boldsymbol{A}$ 的条件数. 在许多情况下, 我们只要条件数大小的一个估计, 而并不需要条件数的精确值. 当前许多求解线性方程组的计算机软件含有估计 $\operatorname{cond}(\boldsymbol{A})$ 的方法, 其中不需要用到 $\boldsymbol{A}^{-1}$. 对矩阵 $\boldsymbol{A}$ 的所谓 1–范数, 即 (6.87) 中的矩阵范数应用于 $\boldsymbol{A}^{\mathrm{T}}$, MATLAB 给出两个用于估计 $\boldsymbol{A}$ 的条件数大小的函数. 调用

```
condA1 = condest(A)
```

产生 1–范数的条件数的估计, 而

```
rcondA1 = rcond(A)
```

则产生条件数估计的倒数. 有了近似条件数, 我们可以大致估计计算解中的误差.

例 6.5.5　一个有名的病态矩阵是希尔伯特 (Hilbert) 矩阵:

$$\boldsymbol{H}_n = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & & \frac{1}{2n-1} \end{bmatrix} \tag{6.92}$$

当 n 增加时, $\boldsymbol{H}_n$ 变得越来越病态. 为了说明病态, 取 $n=4$ 并把 $\boldsymbol{H}_4$ 的元素舍入到 5 位有效数字定义为 $\hat{\boldsymbol{H}}_4$:

$$\hat{\boldsymbol{H}}_4=\begin{bmatrix} 1.000\,0 & 0.500\,00 & 0.333\,33 & 0.250\,00 \\ 0.500\,00 & 0.333\,33 & 0.250\,00 & 0.200\,00 \\ 0.333\,33 & 0.250\,00 & 0.200\,00 & 0.166\,67 \\ 0.250\,00 & 0.200\,00 & 0.166\,67 & 0.142\,86 \end{bmatrix}$$

为了说明 $\boldsymbol{H}_4$ 是病态的, 我们来比较 $\boldsymbol{H}_4^{-1}$ 和 $\hat{\boldsymbol{H}}_4^{-1}$. 它们是方程组 $\boldsymbol{H}_4\boldsymbol{X}=\boldsymbol{I}$ 和 $\hat{\boldsymbol{H}}_4\hat{\boldsymbol{X}}=\boldsymbol{I}$ 的解. 我们有

$$\boldsymbol{H}_4^{-1}=\begin{bmatrix} 16 & -120 & 240 & -140 \\ -120 & 1\,200 & -2\,700 & 1\,680 \\ 240 & -2\,700 & 6\,480 & -4\,200 \\ -140 & 1\,680 & -4\,200 & 2\,800 \end{bmatrix}$$

$$\hat{\boldsymbol{H}}_4^{-1}=\begin{bmatrix} 16.248 & -122.72 & 246.49 & -144.20 \\ -122.72 & 1229.9 & -2771.3 & 1726.1 \\ 246.49 & -2771.3 & 6650.1 & -4310.0 \\ -144.20 & 1726.1 & -4310.0 & 2871.1 \end{bmatrix}$$

$\hat{\boldsymbol{H}}_4$ 中的误差仅存在于第 6 位数字, 但 $\hat{\boldsymbol{H}}_4^{-1}$ 中一些元素的误差则出现于第 2 位数字. 与 $\hat{\boldsymbol{H}}_4$ 中的变化相比, 逆矩阵 $\hat{\boldsymbol{H}}_4^{-1}$ 的变化相当大. 通过直接计算可得

$$\operatorname{cond}(\boldsymbol{H}_4)=\|\boldsymbol{H}_4\|\,\|\boldsymbol{H}_4^{-1}\|=\left(\frac{25}{12}\right)(13\,620)\doteq 28\,000$$

这个结果与 $\hat{\boldsymbol{H}}_4^{-1}$ 中丢失 4 位有效数字这一现象一致. ■

病态矩阵出现在许多应用中. 例如, Hilbert 矩阵自然在函数拟合与数据最小二乘拟合中出现 (参见 4.7 节和 7.1 节), 许多偏微分方程离散化导致中等程度的病态线性方程组和严重的病态线性方程组. 因此, 如果可能, 最好使用有能自动检测病态性质的线性方程组数值求解程序; 不然就利用前面描述的残差校正方法来估计方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 计算解 $\hat{\boldsymbol{x}}$ 的精度.

习题

1. 利用 6.3 节给出的程序 `GEpivot`, 编写求解线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的程序. 要求程序包含以下特点. (i) 产生或读取 $\boldsymbol{A}$ 和 $\boldsymbol{b}$. (ii) 用 `GEpivot` 解 $\boldsymbol{Ax}=\boldsymbol{b}$, 称计算解为 $\hat{\boldsymbol{x}}$. (iii) 计算残差 $\boldsymbol{r}=\boldsymbol{b}-\boldsymbol{A}\hat{\boldsymbol{x}}$, 接着利用 (6.78) 下面描述的过程. (iv) 求解 $\boldsymbol{Ae}=\boldsymbol{r}$ 中误差, 记解为 $\hat{\mathrm{e}}$. (v) 打印 $\hat{\mathrm{e}}$ 和 $\hat{\boldsymbol{x}}+\hat{\mathrm{e}}$. 再打印 $\|\boldsymbol{e}\|/\|\boldsymbol{x}\|$ 的近似值 $\|\hat{\mathrm{e}}\|\,/\,\|\hat{\boldsymbol{x}}\|$. 应用编写的程序求解下列方程组:

(a) $$\begin{cases} 4x_1 - 6x_2 + 4x_3 - x_4 = 0.43 \\ -6x_1 + 14x_2 - 11x_3 + 3x_4 = -1 \\ 4x_1 - 11x_2 + 10x_3 - 3x_4 = 0.82 \\ -x_1 + 3x_2 - 3x_3 + x_4 = -0.23 \end{cases}$$

(b) $$\begin{cases} 5x_1 + 7x_2 + 6x_3 + 5x_4 = 23 \\ 7x_1 + 10x_2 + 8x_3 + 7x_4 = 32 \\ 6x_1 + 8x_2 + 10x_3 + 9x_4 = 33 \\ 5x_1 + 7x_2 + 9x_3 + 10x_4 = 31 \end{cases}$$

它的解是 $x_1 = x_2 = x_3 = x_4 = 1$.

(c) $\boldsymbol{H}_5\boldsymbol{x} = [1, 0.6, 0.4, 0.3, 0.3]^{\mathrm{T}}$, 其中 $\boldsymbol{H}_5$ 由 (6.92) 定义.

2. (a) 对于 (6.87) 定义的向量范数, 证明 (i) $\|\boldsymbol{x}\| = 0$ 的充要条件是 $\boldsymbol{x} = \boldsymbol{0}$, (ii) 对于所有的数 α, $\|\alpha\boldsymbol{x}\| = |\alpha|\,\|\boldsymbol{x}\|$. 对于 (6.87) 定义的矩阵范数也保持同样的性质.

(b) 证明 (6.88) 的范数性质 (P1) 到 (P4).

3. (a) 对于方程组 (6.86), 计算 $\text{cond}(A)$.

(b) 画出方程组 (6.86) 的图形. 方程组右端变化影响的几何意义是什么? 从几何上解释该方程组中等程度病态时的含义.

4. 考虑方程组

$$\begin{cases} 19x_1 + 20x_2 = b_1 \\ 20x_1 + 21x_2 = b_2 \end{cases}$$

计算系数矩阵的条件数. 方程组关于右端常数 $\{b_1, b_2\}$ 的扰动良态吗?

5. 对于

$$\boldsymbol{A} = \begin{bmatrix} 1 & c \\ c & 1 \end{bmatrix}, \quad |c| \neq 1$$

计算 $\text{cond}(\boldsymbol{A})$. 什么时候 A 变为病态? 有关的线性方程组 $Ax = b$ 可说些什么? $\text{cond}(\boldsymbol{A})$ 与 $\det(\boldsymbol{A})$ 的关系怎样?

6. 证明: 对于任何的 $\boldsymbol{A}$, $\text{cond}(\boldsymbol{A}) \geqslant 1$.

提示: 首先证明 $\|\boldsymbol{I}\| = 1$

7. 定义 n 阶矩阵 $\boldsymbol{A}_n$ 为

$$\boldsymbol{A}_n = \begin{bmatrix} 1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & -1 & \cdots & -1 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -1 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

(a) 求 $\boldsymbol{A}_n$ 的逆的显式表示.

提示: 通过求解 $\boldsymbol{B}\boldsymbol{A}_6 = \boldsymbol{I}$, 求 $\boldsymbol{A}_6$ 的逆, 其中 $\boldsymbol{B}$ 为未知的上三角矩阵. 然后, 用得到的结论去 "猜测" 一般的 $\boldsymbol{A}_n$ 的逆.

(b) 计算 cond(A_n).

(c) 对于 $\boldsymbol{b}=[-n+2,-n+3,\cdots,-1,0,1]^{\mathrm{T}}$, $\boldsymbol{A}_n\boldsymbol{x}=\boldsymbol{b}$ 的解是 $\boldsymbol{x}=[1,1,\cdots,1]^{\mathrm{T}}$. 把 $\boldsymbol{b}$ 扰动到 $\hat{\boldsymbol{b}}=\boldsymbol{b}+[0,\cdots,0,\varepsilon]^{\mathrm{T}}$. 求解 $\boldsymbol{A}_n\hat{\boldsymbol{x}}=\hat{\boldsymbol{b}}$ 中的 $\hat{\boldsymbol{x}}$.

提示: 利用 $\hat{\boldsymbol{x}}=\boldsymbol{x}+\boldsymbol{A}_n^{-1}(\hat{\boldsymbol{b}}-\boldsymbol{b})$. 证明 $\boldsymbol{b},\hat{\boldsymbol{b}},\boldsymbol{x},\hat{\boldsymbol{x}}$ 都满足定理 6.5.4.

6.6 迭 代 法

许多应用中出现的线性方程组 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ 的阶数很高. 对于这样的方程组, 6.3 节中介绍的高斯消元法在计算时间或计算机存储需求上常花费太大, 或者两者同时花费太大. 而且, 舍入误差的积累有时使数值解很不正确. 我们通常可以用迭代法求解这种线性方程组, 这是本节的主题.

在迭代法中, 我们产生一系列的近似解逼近方程组的解. 一般地, 即使不考虑舍入误差的影响, 我们也不能期望有限次迭代步能得到正确解. 与之对比, 如果忽略舍入误差, 高斯消元法 $n-1$ 步消元并对得到的上三角方程组回代后就能得到正确解. 因此, 高斯消元法和它的变形通常称为*直接方法*.

在迭代法研究中, 最重要的问题是收敛性. 我们将对一般迭代法提供收敛性分析的框架. 本节研究两种经典迭代方法 —— 雅可比法和高斯–赛德尔法, 并给出收敛的充分条件.

6.6.1 雅可比法和高斯–赛德尔法

我们从一些数值例子入手说明两种流行的迭代方法. 随后讨论更一般的迭代法.

考虑线性方程组

$$\begin{cases} 9x_1+\ \ x_2+\ \ x_3=b_1 \\ 2x_1+10x_2+\ 3x_3=b_2 \\ 3x_1+\ 4x_2+11x_3=b_3 \end{cases} \tag{6.93}$$

求解 (6.93) 的一类迭代方法继续进行如下. 在第 k 个方程中, 将 x_k 用其余的未知量表示. 对于给定的方程组,

$$\begin{cases} x_1=\frac{1}{9}[b_1-x_2-x_3] \\ x_2=\frac{1}{10}[b_2-2x_1-3x_3] \\ x_3=\frac{1}{11}[b_3-3x_1-4x_2] \end{cases} \tag{6.94}$$

记 $\boldsymbol{x}^{(0)}=\left[x_1^{(0)},x_2^{(0)},x_3^{(0)}\right]^{\mathrm{T}}$ 为真解的初始猜测. 对于 $k=0,1,2,\cdots$, 定义迭代序

列:

$$\begin{aligned} x_1^{(k+1)} &= \tfrac{1}{9}\left[b_1 - x_2^{(k)} - x_3^{(k)}\right] \\ x_2^{(k+1)} &= \tfrac{1}{10}\left[b_2 - 2x_1^{(k)} - 3x_3^{(k)}\right] \\ x_3^{(k+1)} &= \tfrac{1}{11}\left[b_3 - 3x_1^{(k)} - 4x_2^{(k)}\right] \end{aligned} \tag{6.95}$$

这个方法称为**雅可比迭代法**或**同步替换法**.

对应于

$$\boldsymbol{b} = [10, 19, 0]^{\mathrm{T}}$$

方程组的解为

$$\boldsymbol{x} = [1, 2, -1]^{\mathrm{T}}$$

表 6-2 给出了解所述方程组雅可比迭代法的数值结果, 在表中,

$$误差 = \left\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\right\|$$

利用了 (6.87) 中定义的范数. 比率表示逐次误差值之比. 浮点算术运算大约包含 16 位十进制数字的精度, 因此, 结果误差正确到表中所示的位数.

表 6-2　求解 (6.93) 的雅可比迭代 (6.95)

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	误差	比率
0	0	0	0	2.00E+0	
1	1.111 1	1.900 0	0	1.00E+0	0.500
2	0.900 0	1.677 8	−0.993 9	3.22E−1	0.322
3	1.035 1	2.018 2	−0.855 6	1.44E−1	0.448
4	0.981 9	1.949 6	−1.016 2	5.04E−2	0.349
5	1.007 4	2.008 5	−0.976 8	2.32E−2	0.462
6	0.996 5	1.991 5	−1.005 1	8.45E−3	0.364
7	1.001 5	2.002 2	−0.996 0	4.03E−3	0.477
8	0.999 3	1.998 5	−1.001 2	1.51E−3	0.375
9	1.000 3	2.000 5	−0.999 3	7.40E−4	0.489
10	0.999 9	1.999 7	−1.000 3	2.83E−4	0.382
30	1.000 0	2.000 0	−1.000 0	3.01E−11	0.447
31	1.000 0	2.000 0	−1.000 0	1.35E−11	0.447

当 k 增加时误差减少; 当 k 越来越大时, 比率值最后趋近约为 0.447 的极限常量. 注意, 尽管误差在减少, 但个别分量的误差可能增大. 例如, 对 $k = 4$ 和 $k = 5$, 比较 $x_3^{(k)}$ 中的误差. 对于许多线性方程组及其求解的迭代法, 比率值会趋向某个极限值, 但在许多情况下, 比率不存在这样的极限.

使用 (6.94) 来产生 (6.93) 的迭代解的另一种方法是尽可能利用每个分量计算时新得到的的信息. 具体地说, 对于 $k=0,1,2,\cdots$, 定义

$$
\begin{aligned}
x_1^{(k+1)} &= \tfrac{1}{9}\left[b_1 - x_2^{(k)} - x_3^{(k)}\right] \\
x_2^{(k+1)} &= \tfrac{1}{10}\left[b_2 - 2x_1^{(k+1)} - 3x_3^{(k)}\right] \\
x_3^{(k+1)} &= \tfrac{1}{11}\left[b_3 - 3x_1^{(k+1)} - 4x_2^{(k+1)}\right]
\end{aligned}
\tag{6.96}
$$

这种方法称为*高斯–赛德尔迭代法*或*逐次替换法*. 这个方法通常比雅可比法更快地收敛.

表 6-3 给出高斯–赛德尔法求解方程组 (6.93) 的迭代次数. 把这些结果与表 6-2 中那些结果作比较. 可以看出, 高斯–赛德尔法的收敛速度比雅可比法 (6.95) 快. 当考虑 k 的值增大时, 比率值似乎并不趋近一个固定的值.

表 6-3 求解 (6.93) 的高斯–赛德尔迭代 (6.96)

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	误差	比率
0	0	0	0	2.00E+0	
1	1.111 1	1.677 8	−0.913 1	3.22E−1	0.161
2	1.026 2	1.968 7	−0.995 8	3.13E−2	0.097
3	1.003 0	1.998 1	−1.000 1	3.00E−3	0.096
4	1.000 2	2.000 0	−1.000 1	2.24E−4	0.074
5	1.000 0	2.000 0	−1.000 0	1.65E−5	0.074
6	1.000 0	2.000 0	−1.000 0	2.58E−6	0.155

6.6.2 一般的迭代格式

为了理解迭代法的特性, 最好按向量–矩阵的形式来讨论. 改写线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 为

$$
\boldsymbol{Nx}=\boldsymbol{b}+\boldsymbol{Px} \tag{6.97}
$$

其中 $\boldsymbol{A}=\boldsymbol{N}-\boldsymbol{P}$ 是 $\boldsymbol{A}$ 的*分裂*. 矩阵 $\boldsymbol{N}$ 必须非奇异, 通常选择它使得线性方程组

$$
\boldsymbol{Nz}=\boldsymbol{f}
$$

比较容易求解. 例如, $\boldsymbol{N}$ 可能是对角矩阵、三角形矩阵或三对角矩阵. 然后, 可以用

$$
\boldsymbol{Nx}^{(k+1)}=\boldsymbol{b}+\boldsymbol{Px}^{(k)}, \quad k=0,1,2,\cdots \tag{6.98}
$$

来定义一个迭代法.

例 6.6.1 (a) 雅可比方法 (6.95) 对应

$$N = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 11 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & -1 & -1 \\ -2 & 0 & -3 \\ -3 & -4 & 0 \end{bmatrix} \tag{6.99}$$

(b) 高斯–赛德尔迭代方法 (6.96) 对应

$$N = \begin{bmatrix} 9 & 0 & 0 \\ 2 & 10 & 0 \\ 3 & 4 & 11 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & -1 & -1 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.100}$$

对于一般的 n 阶矩阵 $A = [a_{ij}]$, 用

$$N = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} \tag{6.101}$$

定义雅可比方法, 而 $P = N - A$. 对于高斯–赛德尔方法, 取

$$N = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{6.102}$$

而 $P = N - A$. 因为雅可比方法中 N 是对角矩阵而高斯–赛德尔方法中 N 是下三角矩阵, 所以对应的线性方程组 $Nz = f$ 都容易求解. 对于能时常应用雅可比方法与高斯–赛德尔方法的方程组 $Ax = b$, 由 (6.101) 或 (6.102) 定义的矩阵 N 非奇异. [例如, 见下面的 (6.107).] ■

为了分析迭代的收敛性, 从 (6.97) 式中减去 (6.98) 式, 并记 $e^{(k)} = x - x^{(k)}$, 得到

$$\begin{aligned} Ne^{(k+1)} &= Pe^{(k)} \\ e^{(k+1)} &= Me^{(k)} \end{aligned} \tag{6.103}$$

其中 $M = N^{-1}P$. 利用 (6.87) 定义的向量和矩阵范数, 得到

$$\|e^{(k+1)}\| \leqslant \|M\|\|e^{(k)}\| \tag{6.104}$$

通过对 k 归纳, 推出

$$\|\boldsymbol{e}^{(k)}\| \leqslant \|\boldsymbol{M}\|^k \|\boldsymbol{e}^{(0)}\| \tag{6.105}$$

因此, 如果

$$\|\boldsymbol{M}\| < 1 \tag{6.106}$$

则误差 $\boldsymbol{e}^{(k)}$ 收敛于零. 我们希望选择分解 $\boldsymbol{A} = \boldsymbol{N} - \boldsymbol{P}$ 使迭代误差收敛到零, 同时方程组 $\boldsymbol{Nz} = \boldsymbol{f}$ 容易求解. 不等式 (6.104) 表明, 每步迭代, 误差至少减少 $\|\boldsymbol{M}\|$ 倍; 如果 (6.106) 成立, 则 (6.105) 表明迭代法对任何初始猜测 $\boldsymbol{x}^{(0)}$ 都收敛.

例 6.6.2 (a) 对方程组 (6.93) 考虑用雅可比方法. 由 (6.99),

$$\boldsymbol{M} = \begin{bmatrix} 0 & -\frac{1}{9} & -\frac{1}{9} \\ -\frac{2}{10} & 0 & -\frac{3}{10} \\ -\frac{3}{11} & -\frac{4}{11} & 0 \end{bmatrix}$$

$$\|\boldsymbol{M}\| = \frac{7}{11} \doteq 0.636$$

这与表 6-2 中 “比率” 列相容, 虽然实际收敛速度比 (6.104) 预计的好.

(b) 对方程组 (6.93) 考虑用高斯–赛德尔方法. 由 (6.100),

$$\boldsymbol{M} = \begin{bmatrix} 0 & -\frac{1}{9} & -\frac{1}{9} \\ 0 & \frac{1}{45} & -\frac{5}{18} \\ 0 & \frac{1}{45} & \frac{13}{99} \end{bmatrix}$$

$$\|\boldsymbol{M}\| = 0.3$$

同样, 这与表 6-3 中的结果相容, 实际收敛速度比 (6.104) 预计的好. ■

对于一般矩阵 $\boldsymbol{A}$ 的雅可比方法, $\boldsymbol{N}$ 如 (6.101) 中那样定义. 我们有

$$\|\boldsymbol{M}\| = \max_{1 \leqslant i \leqslant n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right|$$

条件 (6.106) 等价于要求

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| < |a_{ii}|, \quad 1 \leqslant i \leqslant n \tag{6.107}$$

满足这个条件的矩阵称为**对角占优矩阵**, 对所有这样的线性方程组 $\boldsymbol{Ax} = \boldsymbol{b}$, 雅可比方法都收敛.

对于高斯–赛德尔方法, 利用 (6.102) 不容易计算矩阵 $\boldsymbol{M}$. 但是通过其他方法可以证明, 如果 (6.107) 满足, 则高斯–赛德尔方法的收敛速度与雅可比方法至少一样快, 通常高斯–赛德尔方法收敛速度更快. 见 Atkinson(1989, p.548).

强调一下, (6.107) 仅仅是雅可比方法与高斯–赛德尔方法收敛的充分条件. 对这两种方法存在更好的收敛条件, 但本书不讨论这些条件的详细情况. 有许多线性方程组, 条件 (6.107) 不成立, 但两种迭代法都收敛. 例如, $n \times n$ 矩阵

$$\boldsymbol{A}_n = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix} \tag{6.108}$$

不满足 (6.107), 但是对相应的线性方程组, 雅可比方法与高斯–赛德尔方法都收敛, 这是一个经典结果. 这类矩阵出现在一些微分方程的有限差分解中, 参见 8.8 节.

回到重组的方程组 (6.97), 问题是怎样选择 $\boldsymbol{N}$ 和 $\boldsymbol{P}$ 呢? 一种有益的观点是把它看作用一个 "更简单" 的矩阵 $\boldsymbol{N}$ 逼近 $\boldsymbol{A}$ 的问题. 由于

$$\|\boldsymbol{M}\| = \|\boldsymbol{N}^{-1}\boldsymbol{P}\| \leqslant \|\boldsymbol{N}^{-1}\|\,\|\boldsymbol{P}\|$$

当选择 $\boldsymbol{N}$ 使

$$\|\boldsymbol{N}^{-1}\|\,\|\boldsymbol{P}\| < 1$$

时满足要求 $\|\boldsymbol{M}\| < 1$. 如果用 $\boldsymbol{P} = \boldsymbol{N} - \boldsymbol{A}$, 这可改写成

$$\|\boldsymbol{A} - \boldsymbol{N}\| < \frac{1}{\|\boldsymbol{N}^{-1}\|} \tag{6.109}$$

通常, 选择使这个不等式成立的矩阵 $\boldsymbol{N}$ 是可能的.

对迭代法 (6.98) 可以给出更一般的收敛结论, 包括提供收敛的充要条件. 这样的结果需要用到矩阵本征值概念, 我们在 7.2 节讨论矩阵本征值.

定理 6.6.3 对一切右端 $\boldsymbol{b}$ 和所有初始猜测 $\boldsymbol{x}^{(0)}$, 迭代法 (6.98) 收敛, 当且仅当 $\boldsymbol{M} = \boldsymbol{N}^{-1}\boldsymbol{P}$ 的所有本征值 λ 满足

$$|\lambda| < 1 \tag{6.110}$$

习题 10 讨论这个定理的部分证明. 由 7.2 节习题 9 可知, 对于 $\boldsymbol{M}$ 的所有特征值 λ, $\|\boldsymbol{M}\| < 1$ 蕴含 $|\lambda| < 1$. 因此, 当 $\|\boldsymbol{M}\| < 1$ 时, 条件 (6.110) 满足.

这里应该注意, 迭代法的主要应用之一是求偏微分方程的离散解. 我们可以使用 5.4 节的技巧逼近偏微分方程中未知函数 $u(x, y)$ 的偏导数. 在 xy 平面内的长方形网格点上做这些近似, 就得到关于未知函数 $u(x, y)$ 近似值的一个大型稀疏线性方程组. 细节见 9.1 节. 对于这种问题, 雅可比方法与高斯–赛德尔方法已经被更好和更有效的迭代法代替, 但是这些方法中许多思想类似于这里介绍的方法.

6.6.3 残差校正方法

6.5 节中的方法 (6.82) 到 (6.84) 用于校正高斯消元法得到的解, 现在来推广这个方法. 这也从另一个观点给出上面迭代法的一般模式 (6.97) 到 (6.106).

设 $\boldsymbol{N}$ 是逼近 $\boldsymbol{A}$ 的一个可逆矩阵, 并且设 $\boldsymbol{x}^{(0)}$ 是 $\boldsymbol{Ax}=\boldsymbol{b}$ 解的初始猜测. 定义

$$\boldsymbol{r}^{(0)}=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(0)}$$

由于 $\boldsymbol{x}$ 是 $\boldsymbol{Ax}=\boldsymbol{b}$ 的真解, 所以

$$\boldsymbol{r}^{(0)}=\boldsymbol{Ax}-\boldsymbol{A}\boldsymbol{x}^{(0)}=\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{x}^{(0)})=\boldsymbol{A}\boldsymbol{e}^{(0)}$$

其中 $\boldsymbol{e}^{(0)}=\boldsymbol{x}-\boldsymbol{x}^{(0)}$ 是 $\boldsymbol{x}^{(0)}$ 的误差. 设 $\hat{\mathrm{e}}^{(0)}$ 是

$$\boldsymbol{N}\hat{\mathrm{e}}^{(0)}=\boldsymbol{r}^{(0)}$$

的解, 然后定义

$$\boldsymbol{x}^{(1)}=\boldsymbol{x}^{(0)}+\hat{\mathrm{e}}^{(0)}$$

归纳地重复这个过程:

$$\boldsymbol{r}^{(k)}=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k)} \tag{6.111}$$

$$\boldsymbol{N}\hat{\mathrm{e}}^{(k)}=\boldsymbol{r}^{(k)} \tag{6.112}$$

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\hat{\mathrm{e}}^{(k)} \tag{6.113}$$

这是一般的*残差校正方法*.

这个方法其实就是前面的方法 (6.98), 推导如下:

$$\begin{aligned}\boldsymbol{x}^{(k+1)}&=\boldsymbol{x}^{(k)}+\hat{\mathrm{e}}^{(k)}=\boldsymbol{x}^{(k)}+\boldsymbol{N}^{-1}\boldsymbol{r}^{(k)}\\&=\boldsymbol{x}^{(k)}+\boldsymbol{N}^{-1}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k)})\end{aligned}$$

因此,

$$\begin{aligned}\boldsymbol{N}\boldsymbol{x}^{(k+1)}&=\boldsymbol{N}\boldsymbol{x}^{(k)}+\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k)}\\&=\boldsymbol{b}+(\boldsymbol{N}-\boldsymbol{A})\boldsymbol{x}^{(k)}=\boldsymbol{b}+\boldsymbol{P}\boldsymbol{x}^{(k)}\end{aligned}$$

因为上述推导可逆, 所以这就说明 (6.111~6.113) 与 (6.98) 等价.

有时, 一种迭代形式可能比其他的形式更可取. 例如, 使用 6.5 节的残差校正方法时, 实际产生矩阵 P 会在时间和存储方面代价高昂. 另外, 形式选择依赖于个人的喜好. 迭代法的两种观点在有关文献中是常见的.

MATLAB 程序 这里包含雅可比方法与高斯-赛德尔方法两个样本程序. 一般地, 对于小型方程组, 迭代法不能同选主元的高斯消元法相比. 这里的程序用于测试方法的性能. 在这些程序中, 系数矩阵的特殊结构不加以考虑. 因此, 对于特殊的线性方程组, 应该修正部分有关矩阵 (向量) 乘法的程序. 例如, 对于具有 6.1 节习题 4 和 6.2 节习题 21 中的系数矩阵的线性方程组, 不应该存放系数矩阵, 而可以通过一些标量乘法和加法来实行矩阵 (向量) 乘法.

在求解线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的雅可比方法中, $\boldsymbol{D}$ 表示 $\boldsymbol{A}$ 的对角部分. 迭代公式是

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{D}^{-1}(\boldsymbol{b}-(\boldsymbol{A}-\boldsymbol{D})\boldsymbol{x}^{(k)})$$

根据这种形式实施雅可比方法. 对给定的误差容限 δ, 当最后迭代 $\boldsymbol{x}^{(k+1)}$ 和 $\boldsymbol{x}^{(k)}$ 满足

$$\frac{\left\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\right\|}{\left\|\boldsymbol{x}^{(k+1)}\right\|+\mathsf{eps}}<\delta$$

或达到最大的迭代次数时, 停止迭代. 这里, eps 是机器 ε, 当计算相对误差时常常用它来避免可能除以零. $\|\cdot\|$ 是 (6.87) 中定义的向量范数. 对于向量 $\boldsymbol{x}$, 在 MATLAB 中用

```
norm(x, inf)
```

计算这个范数.

```
function [x, iflag, itnum] = Jacobi(A,b,x0,delta,max_it)
%
% function [x, iflag, itnum] = Jacobi(A,b,x0,delta,max_it)
%
% A program implementing the Jacobi iteration method to
% solve the linear system Ax=b.
%
% Input
% A: square coefficient matrix.   It is modified on output
% b:  right side vector
% x0:  initial guess
% delta:  error tolerance for the relative difference
%          between two consecutive iterates
% max_it:  maximum number of iterations to be allowed
%
% Output
% x:  numerical solution vector
% iflag:  1 if a numerical solution satisfying the error
%          tolerance is found within max_it iterations
%       -1 if the program fails to produce a numerical
%          solution in max_it iterations
% itnum:  the number of iterations used to compute x
```

```
% initialization
n = length(b);
iflag = 1;
k = 0;

% create a vector with diagonal elements of A
diagA = diag(A);
% modify A to make its diagonal elements zero
A = A-diag(diagA);

% iteration
while k < max_it
  k = k+1;
  x = (b-A*x0)./diagA;
  relerr = norm(x-x0,inf)/(norm(x,inf)+eps);
  x0 = x;
  if relerr < delta
    break
  end
end

itnum = k;
if itnum == max_it
  iflag = -1
end
```

对于高斯–赛德尔方法, 迭代公式是

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \quad 1 \leqslant i \leqslant n$$

注意可以由行向量和列向量之积来计算两个和式.

```
function [x, iflag, itnum] = GS(A,b,x0,delta,max_it)
%
% function [x, iflag, itnum] = GS(A,b,x0,delta,max_it)
%
% A program implementing the Gauss-Seidel iteration
% method to solve the linear system Ax=b.
%
% Input
% A: square coefficient matrix
% b:  right side vector
% x0:  initial guess
% delta:  error tolerance for the relative difference
% between two consecutive iterates
```

```
% max_it:  maximum number of iterations to be allowed
%
% Output
% x:  numerical solution vector
% iflag:  1 if a numerical solution satisfying the error
%         tolerance is found within max_it iterations
%        -1 if the program fails to produce a numerical
%          solution in max_it iterations
% itnum:  the number of iterations used to compute x

% initialization
n = length(b);
iflag = 1;
k = 0;
x = x0;

% iteration
while k < max_it
  k = k+1;
  x(1) = (b(1)-A(1,2:n)*x0(2:n))/A(1,1);
  for i = 2:n-1
    x(i) = (b(i)-A(i,1:i-1)*x(1:i-1) ...
           -A(i,i+1:n)*x0(i+1:n))/A(i,i);
  end
  x(n) = (b(n)-A(n,1:n-1)*x(1:n-1))/A(n,n);
  relerr = norm(x-x0,inf)/(norm(x,inf)+eps);
  x0 = x
  if relerr < delta
    break
  end
end

itnum = k;
if itnum == max_it
  iflag = -1
end
```

习题

1. 对下面选择的右端 b 和真解 x, 使用雅可比和高斯–赛德尔迭代法求方程组 (6.93) 的迭代解. 使用 $\boldsymbol{x}^{(0)} = [0 \quad 0 \quad 0]^{\mathrm{T}}$. 迭代到 $\left\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\right\| \leqslant 0.000\,05$ 为止. 如同表 6-2 和表 6-3 那样, 考虑逐次误差之比.
   (a) $\boldsymbol{b} = [16 \quad 44 \quad 59]^{\mathrm{T}}, \quad \boldsymbol{x} = [1 \quad 3 \quad 4]^{\mathrm{T}}$
   (b) $\boldsymbol{b} = [0 \quad -7 \quad 7]^{\mathrm{T}}, \quad \boldsymbol{x} = [0 \quad -1 \quad 1]^{\mathrm{T}}$
2. 对于下列方程组重复习题 1 的工作:

$$A = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 8 & 1 & 1 \\ 1 & 1 & 7 & 1 \\ 1 & 1 & 1 & 6 \end{bmatrix}, \quad b = \begin{bmatrix} 75 \\ 54 \\ 43 \\ 34 \end{bmatrix}, \quad x = \begin{bmatrix} 7 \\ 5 \\ 4 \\ 3 \end{bmatrix}$$

3. 考虑线性方程组

$$x_1 + 4x_2 = 1$$

$$4x_1 + x_2 = 0$$

真解是 $x_1 = -1/15$, $x_2 = 4/15$. 对该方程组取 $x^{(0)} = [0\ ,\ 0]^{\mathrm{T}}$, 应用雅可比和高斯–赛德尔迭代法求解, 并且找出哪种方法发散更快. 随后, 交换两个方程将方程组写成

$$4x_1 + x_2 = 0$$

$$x_1 + 4x_2 = 1$$

并且取 $x^{(0)} = [0\ ,\ 0]^{\mathrm{T}}$, 再应用两种方法求解. 迭代到 $\left\|x - x^{(k)}\right\| \leqslant 10^{-5}$ 为止. 哪种方法收敛更快?

4. 对于

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ -3 \\ -7 \\ 9 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 2 \end{bmatrix}$$

考虑方程组 $Ax = b$ 的迭代解.

(a) 使用雅可比方法, 取 $x^{(0)} = \mathbf{0}$. 迭代到 $\left\|x - x^{(k)}\right\| \leqslant 0.000\,05$ 为止. 观察逐次误差之比. 计算 $\|M\|$.

(b) 使用高斯–赛德尔方法重复 (a) 中的工作. 这里, 计算 $\|M\|$ 比 (a) 中的工作困难一些.

5. 考虑线性方程组

$$\begin{bmatrix} 4 & 1 & 1 \\ 1 & 2 & 0 \\ 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

(a) 对于任何右端向量 $b = [b_1, b_2, b_3]^{\mathrm{T}}$, 证明该方程组有唯一解 $x = [x_1, x_2, x_3]^{\mathrm{T}}$.

(b) 当对这方程组应用雅可比方法或高斯–赛德尔方法时, 你预期迭代收敛吗? 请给出解释.

6. 考虑由 (6.108) 给出的系数矩阵 A_n 的线性方程组. 对于 $n = 5, 10, 20$, 应用定理 6.6.3, 并使用 MATLAB 计算迭代矩阵 M 的本征值, 探究雅可比方法和高斯–赛德尔方法的收敛性.

提示: MATLAB 中使用 `eig(B)` 可得到矩阵 B 的全部本征值.

7. 考虑求解方程组

$$A_n x = b_n \tag{6.114}$$

其中 $\boldsymbol{A}_n$ 在 (6.108) 中给出并且

$$\boldsymbol{b}_n = \frac{1}{(n+1)^2}[1,\cdots,1]^{\mathrm{T}}$$

取 $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, 使用雅可比方法和高斯–赛德尔方法. 直到两次相邻迭代之差小于 ε, 即直到 $\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right\| \leqslant \varepsilon$ 为止. 对 $n = 10, 100, 1000$ 和 $\varepsilon = 10^{-6}, 10^{-8}, 10^{-10}$ 进行计算. 给出完成计算的整数 k 的表格. 根据表格中给出的结果评论两种方法的特性.

8. 继续习题 7, 对于固定的 n, 将解写成

$$\boldsymbol{x} = [x_1,\cdots,x_n]^{\mathrm{T}}$$

对于 $1 \leqslant i \leqslant n$, 可以证明 x_i 接近于 $t_i(1-t_i)/2$, $t_i = i/(n+1)$. 对于每个 n 和每个 ε, $\boldsymbol{x}^{(k)}$ 表示习题 7 中计算的数值解, 画出误差向量

$$\boldsymbol{e} = [e_1,\cdots,e_n]^{\mathrm{T}}$$
$$e_i = x_i^{(k)} - \frac{1}{2}t_i(1-t_i), \quad 1 \leqslant i \leqslant n$$

关于向量 $\boldsymbol{t} = [t_1,\ldots,t_n]^{\mathrm{T}}$ 的图形. 计算 $\|\boldsymbol{e}\|$ 并观察 n 增加时 $\|\boldsymbol{e}\|$ 如何减少?

9. 对下面给出的方程组重复习题 4 的工作. 为了计算高斯–赛德尔方法中的矩阵 $\boldsymbol{M} = \boldsymbol{N}^{-1}\boldsymbol{P}$, 需要使用程序先计算 $\boldsymbol{N}^{-1}$ 再用 $\boldsymbol{P}$ 相乘. 另外, 利用定理 6.6.3 作为检验收敛性的另一种方法. 因此, 必须计算 $\boldsymbol{M}$ 的最大本征值.

$$\boldsymbol{A} = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} 4 \\ -1 \\ -5 \\ -2 \\ 2 \\ 2 \\ -1 \\ 1 \\ 6 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

这个方程组来自求解偏微分方程

$$\frac{\partial^2 x(s,t)}{\partial s^2}+\frac{\partial^2 x(s,t)}{\partial t^2}=f(s,t), \quad 0\leqslant s,t\leqslant 1$$

10. (a) 对于迭代法 (6.98), 证明

$$\boldsymbol{e}^{(k)}=\boldsymbol{M}^k\boldsymbol{e}^{(0)}, \quad k\geqslant 0, \quad \boldsymbol{M}=\boldsymbol{N}^{-1}\boldsymbol{P}$$

上述等式表明, 当 $k\to\infty$ 时, 为了对所有选择的 $\boldsymbol{e}^{(0)}$, 误差 $\boldsymbol{e}^{(k)}$ 趋向于零, 我们必须有 $\boldsymbol{M}^k\to\boldsymbol{0}$ (零矩阵).

(b) 证明: 如果对所有的初始猜测 $\boldsymbol{x}^{(0)}$ 或等价地对所有的 $\boldsymbol{e}^{(0)}$, 迭代法 (6.98) 收敛, 则矩阵 $\boldsymbol{M}=\boldsymbol{N}^{-1}\boldsymbol{P}$ 的所有本征值 λ 满足 $|\lambda|<1$.

提示: 证明这个命题的逆否命题. 假定 $\boldsymbol{M}$ 的某个本征值 λ 满足 $|\lambda|\geqslant 1$. 证明对于某个选择的 $\boldsymbol{e}^{(0)}$, $\boldsymbol{e}^{(k)}$ 不趋向于零, 以 λ 相应的本征向量为基础考虑 $\boldsymbol{e}^{(0)}$ 的选择.

11. 许多线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 可直接写成 $\boldsymbol{x}=\boldsymbol{b}+\boldsymbol{Mx}$ 的形式, 其中 $\boldsymbol{A}=\boldsymbol{I}-\boldsymbol{M}$. 利用本节的思想方法对

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{b}+\boldsymbol{M}\boldsymbol{x}^{(k)}, \quad k\geqslant 0$$

作收敛性分析.

12. 考虑迭代

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{b}+\alpha\begin{bmatrix}2&1\\1&2\end{bmatrix}\boldsymbol{x}^{(k)}, \quad k=0,1,2,\cdots$$

其中 α 是实常量. 对于某些 α 值和任何选择的初始猜测, 迭代法收敛; 而对于 α 的其他值, 这个方法发散. 找出使这个方法收敛的所有 α 值.

13. 利用习题 11 的迭代法, 解线性方程组 $\boldsymbol{x}=\boldsymbol{b}+\boldsymbol{Mx}$, 其中

$$\boldsymbol{M}_{ij}=\frac{1}{2n}\left[\frac{t_i^3}{1+t_j}+1\right], \quad b_i=\frac{1}{4}+t_i-\frac{1}{2}t_i^3$$

并且 $t_i=(2i-1)/2n$, $1\leqslant i\leqslant n$. 真解是

$$x_i=1+t_i, \quad 1\leqslant i\leqslant n$$

对几个 n 值, 例如 $n=3,6,12,24$ 或更大的值, 求解相应的方程组. 计算误差 $\left\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\right\|$ 和它们递减的比率. 这样的方程组来自下列积分方程

$$x(s)-\frac{1}{2}\int_0^1\left[\frac{s^3}{1+t}+1\right]x(t)\mathrm{d}t=\frac{1}{4}+s-\frac{1}{2}s^3, \quad 0\leqslant s\leqslant 1$$

的数值求解.

14. 考虑求解线性方程组 $\boldsymbol{A}_\varepsilon\boldsymbol{x}=\boldsymbol{b}$, 其中 $\boldsymbol{A}_\varepsilon=\boldsymbol{A}_0+\varepsilon\boldsymbol{B}$,

$$\boldsymbol{A}_0=\begin{bmatrix}2&1&0\\1&2&1\\0&1&2\end{bmatrix}, \quad \boldsymbol{B}=\begin{bmatrix}0&1&1\\-1&0&1\\-1&-1&0\end{bmatrix}$$

对于各种 ε 值和 $\boldsymbol{N}=\boldsymbol{A}_0$, 使用残差校正法 (6.111~6.113) 求解 $\boldsymbol{A}_\varepsilon\boldsymbol{x}=\boldsymbol{b}$. 对于某个 α, 给出 $|\varepsilon|\leqslant\alpha$ 形式成立的条件, 确保残差校正法收敛.

第 7 章　数值线性代数：续篇

本章中我们继续讨论数值线性代数中的一些问题. 7.1 节定义最小二乘数据拟合的方法, 导出众所周知的一些线性方程组. 7.2 节介绍矩阵本征值问题, 以及关于它的解的一种数值方法. 7.3 节讨论非线性方程组, 并推广了牛顿法求解这些方程组.

7.1　最小二乘数据拟合

在科学和工程实验中, 物理量的测量一般不太准确. 除了可能是人为误差造成的之外, 常常是由于测量所用的设备固有限制所造成的. 本节中我们给出一种方法, 得到变量之间的关系, 并使这些变量中的测量误差造成的影响达到最小.

假设一个实验涉及测量两个相关的变量 x 和 y. 我们的目的是得到这两个变量之间的关系:

$$y = f(x) \tag{7.1}$$

在实验中, 选择 x 的一些值, 例如说, $x_1, x_2, \cdots, x_n$, 然后测量 y 对应的值. 设用 $y_1, \cdots, y_n$ 表示实际的测量值, 并用

$$\varepsilon_i = f(x_i) - y_i$$

表示测量误差. 实验员希望利用点 $(x_1, y_1), \cdots, (x_n, y_n)$ 尽可能精确地确定解析关系式 (7.1).

利用多项式插值、样条函数或其他形式的插值函数总可以得到解析关系 (7.1). 但这些方法忽略误差 $\{\varepsilon_i\}$ 的存在, 常常得出对函数 $f(x)$ 糟糕的估计. 实验员通常知道或猜测未知函数 $f(x)$ 属于某个函数类, 例如, 多项式函数类. 然后就可选择那个函数类中的函数来最好地逼近未知函数 $f(x)$, 并考虑实验误差 $\{\varepsilon_i\}$ 的影响. 作为这种情况的一个例子, 考虑表 7-1 中的数据并在图 7-1 中画出它的图形. 由这个图形知, 可以期望 $f(x)$ 接近于一个线性多项式

$$f(x) = mx + b \tag{7.2}$$

如果假设这是个正确的形式, 那么现在确定 $f(x)$ 的问题归结为决定常数 m 和 b.

表 7-1 经验数据

x_i	y_i	x_i	y_i
1.0	−1.945	3.2	0.764
1.2	−1.253	3.4	0.532
1.4	−1.140	3.6	1.073
1.6	−1.087	3.8	1.286
1.8	−0.760	4.0	1.502
2.0	−0.682	4.2	1.582
2.2	−0.424	4.4	1.993
2.4	−0.012	4.6	2.473
2.6	−0.190	4.8	2.503
2.8	0.452	5.0	2.322
3.0	0.337		

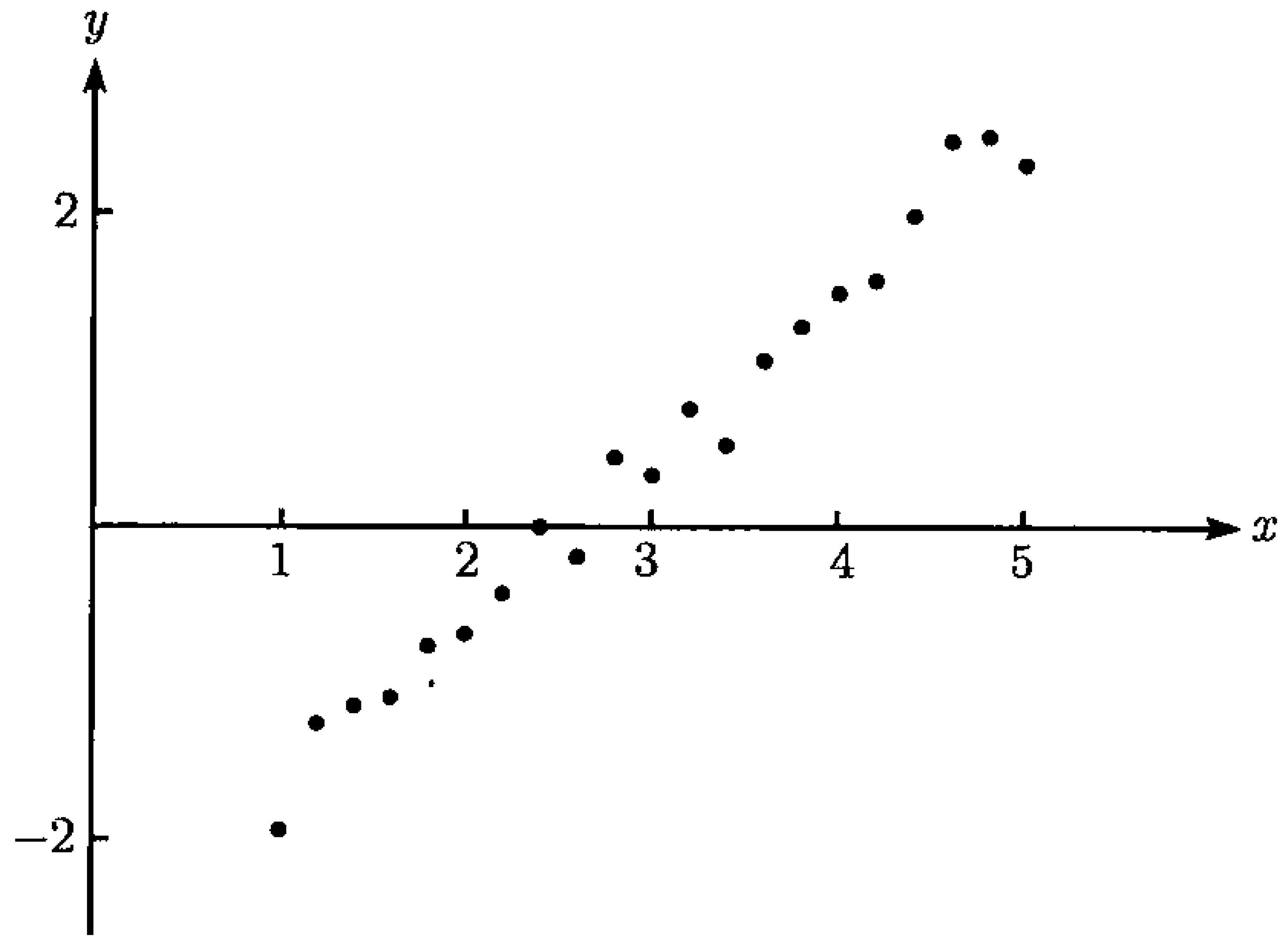


图 7-1 表 7-1 中的经验数据的图形

为了确定 $f(x)$ 合理的近似, 我们必须假设知道更多有关误差 $\{\varepsilon_i\}$ 的信息. 一种标准的假设是每个误差为服从*正态概率分布*的随机变量. 这是概率论和统计课程中讨论的一个论题, 这里不予考虑. 满足这个假设的误差有下列一般性质. (1) 如果对相同的 $x = x_i$ 重复许多次实验, 则实验值 y_i 所相应的未知误差 ε_i 的可能平均值为零; (2) 对 $x = x_i$ 这个相同的实验情况, 当 $|\varepsilon_i|$ 增加时, 误差发生的可能性将迅速地减少. 有很好的理由认为误差 $\{\varepsilon_i\}$ 满足这些性质, 我们在下面称这样的误差服从*正态误差假设*. 为了进一步简化下面的讨论, 我们假设每个误差 ε_i, $1 \leqslant i \leqslant n$, 都是服从同样正态概率分布函数的随机变量, 这意味 ε_i 的大小与 x_i 或 y_i 的大小无关.

假设函数 $f(x)$ 属于某个函数类 C. 例如, 对表 7-1 中的数据假设 $f(x)$ 是线性的. 对 C 中所有的函数 $\hat{f}(x)$, 可以证明, 最有可能等于 f 的函数 $\hat{f}^*$ 也对 C 中所有的函数 $\hat{f}$ 使表达式

$$E=\sqrt{\frac{1}{n}\sum_{i=1}^{n}[\hat{f}(x_i)-y_i]^2} \tag{7.3}$$

达到极小. 此式称为用 $\hat{f}(x)$ 逼近数据 $\{y_i\}$ 的**均方根误差**. 对 C 中所有 $\hat{f}$ 极小化 E 得到的函数 $\hat{f}^*(x)$ 称为数据 $\{(x_i,y_i)\}$ 的**最小二乘逼近**.

7.1.1 线性最小二乘逼近

我们考虑在 E 中使用 $\hat{f}(x)=mx+b$. 对于这样的函数 $\hat{f}(x)$, 在试图极小化 (7.3) 之前, 注意到极小化 E 等价于极小化 (7.3) 中的和, 尽管极小值会不同. 我们求使

$$G(b,m)=\sum_{i=1}^{n}[mx_i+b-y_i]^2 \tag{7.4}$$

达到极小的 b 和 m, 它们同样使 E 达到极小.

由多元微积分知道, 使 $G(b,m)$ 极小化的 (b,m) 值满足

$$\frac{\partial G(b,m)}{\partial b}=0,\quad \frac{\partial G(b,m)}{\partial m}=0 \tag{7.5}$$

由 (7.4) 得

$$\frac{\partial G}{\partial b}=\sum_{i=1}^{n}2(mx_i+b-y_i)$$

$$\frac{\partial G}{\partial m}=\sum_{i=1}^{n}2(mx_i+b-y_i)x_i=\sum_{i=1}^{n}2(mx_i^2+bx_i-x_iy_i)$$

上式与 (7.5) 式结合, 得到 2 阶线性方程组

$$\begin{aligned}nb+\left(\sum_{i=1}^{n}x_i\right)m&=\sum_{i=1}^{n}y_i\\ \left(\sum_{i=1}^{n}x_i\right)b+\left(\sum_{i=1}^{n}x_i^2\right)m&=\sum_{i=1}^{n}x_iy_i\end{aligned} \tag{7.6}$$

倘若系数行列式

$$n\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2\neq 0 \tag{7.7}$$

非零, 则方程组有唯一解 (b,m). 除了平凡情况

$$x_1=x_2=\cdots=x_n=\text{常量}$$

之外, 这个条件应该成立. 因为假设数 $\{x_j\}$ 不同, 所以这里不会出现平凡情况.

例 7.1.1 对表 7-1 中的数据确定最小二乘拟合. 我们有 $n=21$, 并且

$$\sum_{i=1}^{n} x_i = 63.0, \qquad \sum_{i=1}^{n} x_i^2 = 219.8$$
$$\sum_{i=1}^{n} y_i = 9.326, \quad \sum_{i=1}^{n} x_i y_i = 60.730\ 2$$

在 (7.6) 中使用这些值, 然后求解 b 和 m, 得到

$$b \doteq -2.746\ 05, \quad m \doteq 1.063\ 38$$

表 7-1 中的数据的线性最小二乘拟合是

$$\hat{f}^*(x) = 1.063\ 38x - 2.746\ 05 \tag{7.8}$$

在所有线性函数 $\hat{f}(x) = mx + b$ 中, 在使 (7.3) 中均方根误差 E 达到极小的意义下, (7.8) 是最优的. 关于 (7.8), 相应的 E 的最小值是

$$E \doteq 0.171$$

它可以作为用最小二乘拟合 (7.8) 逼近表 7-1 中数据的平均误差. (7.8) 的图形和表 7-1 中数据点在图 7-2 给出.

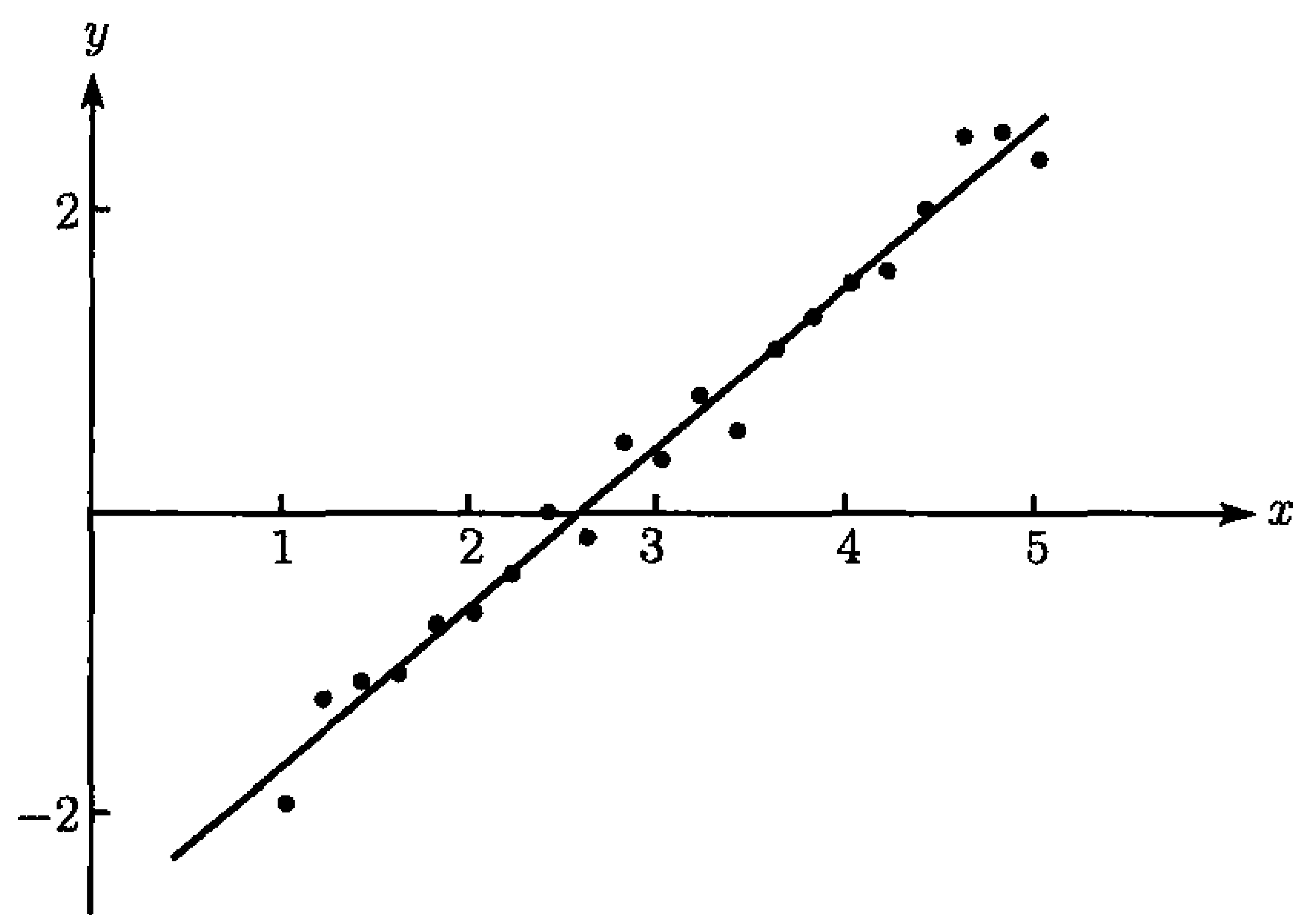


图 7-2 (7.8) 的最小二乘拟合 $\hat{f}^*(x)$

7.1.2 多项式最小二乘逼近

大多数数据不对应线性函数, 而对应更复杂的函数. 设用来表示给定的数据 $\{(x_i, y_i)\}$ 的函数 $\hat{f}(x)$ 为

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \cdots + a_m\varphi_m(x) \tag{7.9}$$

其中 $a_1, a_2, \ldots, a_m$ 为任意数, 而 $\varphi_1(x), \ldots, \varphi_m(x)$ 是已知函数. 例如, 若 $\hat{f}(x)$ 是二次多项式, 则可以记

$$\hat{f}(x) = a_1 + a_2 x + a_3 x^2 \tag{7.10}$$

其中

$$\varphi_1(x) \equiv 1, \quad \varphi_2(x) = x, \quad \varphi_3(x) = x^2$$

在正态误差假设下, 选择函数 $\hat{f}(x)$ 使均方根误差 (7.3) 极小化. 我们以 $m = 3$ 为例:

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + a_3\varphi_3(x)$$

并选择常数 a_1, a_2, a_3 使表示式

$$G(a_1, a_2, a_3) = \sum_{j=1}^{n} [a_1\varphi_1(x_j) + a_2\varphi_2(x_j) + a_3\varphi_3(x_j) - y_j]^2$$

达到极小. 使 $G(a_1, a_2, a_3)$ 达到极小的点 (a_1, a_2, a_3) 满足

$$\frac{\partial G}{\partial a_1} = 0, \quad \frac{\partial G}{\partial a_2} = 0, \quad \frac{\partial G}{\partial a_3} = 0$$

由此导出三个方程,

$$0 = \frac{\partial G}{\partial a_i} = \sum_{j=1}^{n} 2[a_1\varphi_1(x_j) + a_2\varphi_2(x_j) + a_3\varphi_3(x_j) - y_j]\varphi_i(x_j)$$

其中 $i = 1, 2, 3$. 化简, 得到

$$\begin{aligned}\left[\sum_{j=1}^{n}\varphi_1(x_j)\varphi_i(x_j)\right] a_1 &+ \left[\sum_{j=1}^{n}\varphi_2(x_j)\varphi_i(x_j)\right] a_2 \\ &+ \left[\sum_{j=1}^{n}\varphi_3(x_j)\varphi_i(x_j)\right] a_3 = \sum_{j=1}^{n} y_j\varphi_i(x_j), \quad i = 1, 2, 3\end{aligned} \tag{7.11}$$

这是关于三个未知量 a_1, a_2, a_3 的 3 个线性方程.

例 7.1.2 对于 (7.10) 式中的二次多项式 $f(x)$ 应用 (7.11) 式, 得到的三个方程是

$$na_1 + \left[\sum_{j=1}^{n} x_j\right] a_2 + \left[\sum_{j=1}^{n} x_j^2\right] a_3 = \sum_{j=1}^{n} y_j$$

$$\left[\sum_{j=1}^{n} x_j\right] a_1 + \left[\sum_{j=1}^{n} x_j^2\right] a_2 + \left[\sum_{j=1}^{n} x_j^3\right] a_3 = \sum_{j=1}^{n} y_j x_j \tag{7.12}$$
$$\left[\sum_{j=1}^{n} x_j^2\right] a_1 + \left[\sum_{j=1}^{n} x_j^3\right] a_2 + \left[\sum_{j=1}^{n} x_j^4\right] a_3 = \sum_{j=1}^{n} y_j x_j^2$$

由于假设点 $\{x_i\}$ 相异, 可以证明这是一个非奇异线性方程组. ■

当 $\hat{f}(x)$ 取 (7.9) 中给出的更一般形式时, 由线性方程组

$$\sum_{k=1}^{m} a_k \left[\sum_{j=1}^{n} \varphi_k(x_j)\varphi_i(x_j)\right] = \sum_{j=1}^{n} y_j \varphi_i(x_j), \quad i = 1, \cdots, m \tag{7.13}$$

确定使 (7.3) 均方根误差 E 极小化的系数 $a_1, \cdots, a_m$. 其推导留作习题 9. $\hat{f}(x)$ 是 $m-1$ 次多项式的特殊情况非常重要, 我们来仔细地加以讨论.

把 $\hat{f}(x)$ 写成 $m-1$ 次多项式的标准形式

$$\hat{f}(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_m x^{m-1}$$

然后把它与 (7.9) 对应, 可得

$$\varphi_1(x) = 1, \quad \varphi_2(x) = x, \quad \varphi_3(x) = x^2, \quad \cdots, \quad \varphi_m(x) = x^{m-1} \tag{7.14}$$

确定 $\{a_j\}$ 的方程组 (7.13) 变为

$$\sum_{k=1}^{m} a_k \left[\sum_{j=1}^{n} x_j^{i+k-2}\right] = \sum_{j=1}^{n} y_j x_j^{i-1}, \quad i = 1, 2, \cdots, m \tag{7.15}$$

当 $m = 3$ 时, 给出前面得到的方程组 (7.12).

对 $m < n$, 方程组 (7.15) 非奇异, 但遗憾的是, 当 $\hat{f}(x)$ 的次数增加时方程组变得非常病态. 实际上, 即使对较小的 m 值 (如 $m = 4$), 系数矩阵的条件数也已经相当大. 因此, 用 (7.14) 作最小二乘多项式拟合一般不可取, 除非次数 $\leqslant 2$.

用更高次多项式 $\hat{f}(x)$ 对数据 $\{(x_i, y_i)\}$ 作最小二乘拟合, 必须将它写成

$$\hat{f}(x) = a_1\varphi_1(x) + \cdots + a_m\varphi_m(x)$$

的形式, 并选择 $\varphi_1(x), \cdots, \varphi_m(x)$ 使 (7.13) 中的系数矩阵不病态. 事实上, 存在这些函数 $\varphi_j(x)$ 的最优选择, 其次数 $\deg(\varphi_j) = j - 1$ 并且 (7.13) 的系数矩阵为对角阵. 非最优的但一般还是令人满意的选择可以是 4.5 节的切比雪夫多项式 $\{T_k(x)\}$, 稍许更好一点的选择是 4.7 节的勒让德多项式.

假设从区间 $[\alpha,\beta]$ 内选择结点 $\{x_i\}$. 引入修正的切比雪夫多项式

$$\varphi_k(x)=T_{k-1}\left(\frac{2x-\alpha-\beta}{\beta-\alpha}\right),\quad \alpha\leqslant x\leqslant\beta,\quad k\geqslant 1 \tag{7.16}$$

则次数 $\deg(\varphi_k)=k-1$, 并且任何 $m-1$ 次多项式 $\hat{f}(x)$ 可写成 $\varphi_1(x),\cdots,\varphi_m(x)$ 的线性组合.

例 7.1.3　考虑表 7-2 中的数据并在图 7-3 中绘出它的图形. 我们先用正规表示式

$$\hat{f}(x)=a_1+a_2x+a_3x^2+a_4x^3 \tag{7.17}$$

求 $\hat{f}(x)$ 的三次多项式最小二乘拟合.

表 7-2　三次多项式最小二乘拟合的数据

x_i	y_i	x_i	y_i
0.00	0.486	0.55	1.102
0.05	0.866	0.60	1.099
0.10	0.944	0.65	1.017
0.15	1.144	0.70	1.111
0.20	1.103	0.75	1.117
0.25	1.202	0.80	1.152
0.30	1.166	0.85	1.265
0.35	1.191	0.90	1.380
0.40	1.124	0.95	1.575
0.45	1.095	1.00	1.857
0.50	1.122		

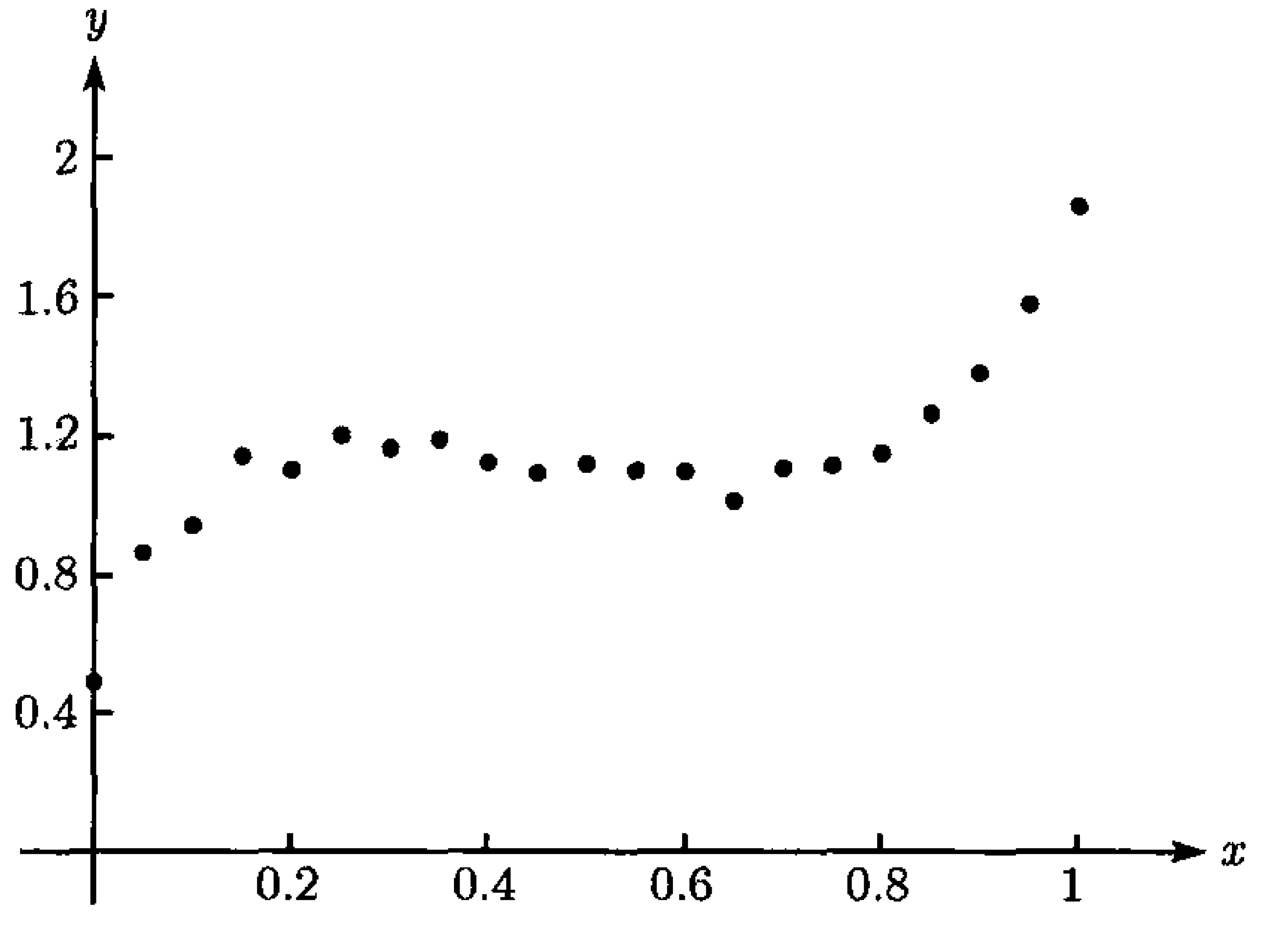


图 7-3　表 7-2 中数据的图形

所得的线性方程组 (7.15) 用 $\boldsymbol{La}=\boldsymbol{b}$ 表示, 其中

$$
\boldsymbol{L}=\begin{bmatrix} 21 & 10.5 & 7.175 & 5.512\,5 \\ 10.5 & 7.175 & 5.512\,5 & 4.516\,66 \\ 7.175 & 5.512\,5 & 4.516\,66 & 3.854\,16 \\ 5.512\,5 & 4.516\,66 & 3.854\,16 & 3.382\,12 \end{bmatrix}
$$
$$
\boldsymbol{a}=[a_1,a_2,a_3,a_4]^{\mathrm{T}}
$$
$$
\boldsymbol{b}=[24.118\,0,13.234\,5,9.468\,36,7.559\,44]^{\mathrm{T}}
$$

方程组的解是

$$
\boldsymbol{a}=[0.574\,7,4.725\,9,-11.128\,2,7.668\,7]^{\mathrm{T}}
$$

可以算出 (6.91) 中定义的条件数为

$$
\operatorname{cond}(L)=\|L\|\|L^{-1}\|\doteq 22\,000 \tag{7.18}
$$

这是个非常大的值, 它表示求 $\boldsymbol{La}=\boldsymbol{b}$ 精确解可能很困难. 为了验证这个结论, 对上面的 $\boldsymbol{b}$ 加上扰动

$$
[0.01,-0.01,0.01,-0.01]^{\mathrm{T}}
$$

由于数据误差, 这将改变 $\boldsymbol{b}$ 中小数右边的第 2 位. 新的扰动方程组的解是

$$
\boldsymbol{a}=[0.740\,8,2.682\,5,-6.153\,8,4.455\,0]^{\mathrm{T}}
$$

这与前面的 a 的结果显著不同. 这里关键是使用 (7.17) 导致了一个相当病态的线性方程组. 为了用性态较好的线性方程组对表 7-2 的数据求三次多项式的最小二乘逼近, 在 $[0,1]$ 上用 (7.16) 中的函数并记

$$
f(x)=a_1\varphi_1(x)+a_2\varphi_2(x)+a_3\varphi_3(x)+a_4\varphi_4(x) \tag{7.19}
$$

其中

$$
\begin{aligned}
&\varphi_1(x)=T_0(2x-1)\equiv 1\\
&\varphi_2(x)=T_1(2x-1)=2x-1\\
&\varphi_3(x)=T_2(2x-1)=2(2x-1)^2-1=8x^2-8x+1\\
&\varphi_4(x)=T_3(2x-1)=4(2x-1)^3-3(2x-1)=32x^3-48x^2+18x-1
\end{aligned}
$$

由于在 (7.19) 的 $\hat{f}(x)$ 的表示式中使用了不同的函数, a_1,a_2,a_3,a_4 与表示式 (7.17)

中的完全不同. 仍用 $\boldsymbol{L}\boldsymbol{a}=\boldsymbol{b}$ 表示线性方程组 (7.13), 则

$$\boldsymbol{L}=\begin{bmatrix} 21 & 0 & -5.6 & 0 \\ 0 & 7.7 & 0 & -2.833\,6 \\ -5.6 & 0 & 10.466\,4 & 0 \\ 0 & -2.833\,6 & 0 & 11.010\,56 \end{bmatrix}$$

$$\boldsymbol{b}=[24.118, 2.351, -6.011\,08, 1.523\,576]^{\mathrm{T}}$$

方程组的解是

$$\boldsymbol{a}=[1.160\,969, 0.393\,514, 0.046\,850, 0.239\,646]^{\mathrm{T}}$$

线性方程组关于前面使用 (7.17) 的三次多项式最小二乘拟合以及 $\boldsymbol{b}$ 中的那类扰动非常稳定. 这个结论也可由 $\boldsymbol{L}$ 的条件数较小来得到:

$$\operatorname{cond}(\boldsymbol{L})=\|\boldsymbol{L}\|\|\boldsymbol{L}^{-1}\|\doteq(26.6)(0.1804)\doteq 4.8 \tag{7.20}$$

由 6.5 节的结论 (6.89), 我们知道 $\boldsymbol{b}$ 中相对小的扰动将导致解 a 相应的小变化. 再把 (7.20) 与前面方程组的条件数 (7.18) 作比较. 为了给出逼近表 7-2 中数据的 $\hat{f}(x)$ 的精度, 从 (7.3) 容易计算均方根误差为

$$E\doteq 0.042\,1$$

与 $\hat{f}(x)$ 的函数值相比, 这是一个相当小的值. $\hat{f}(x)$ 的图形如图 7-4 所示.

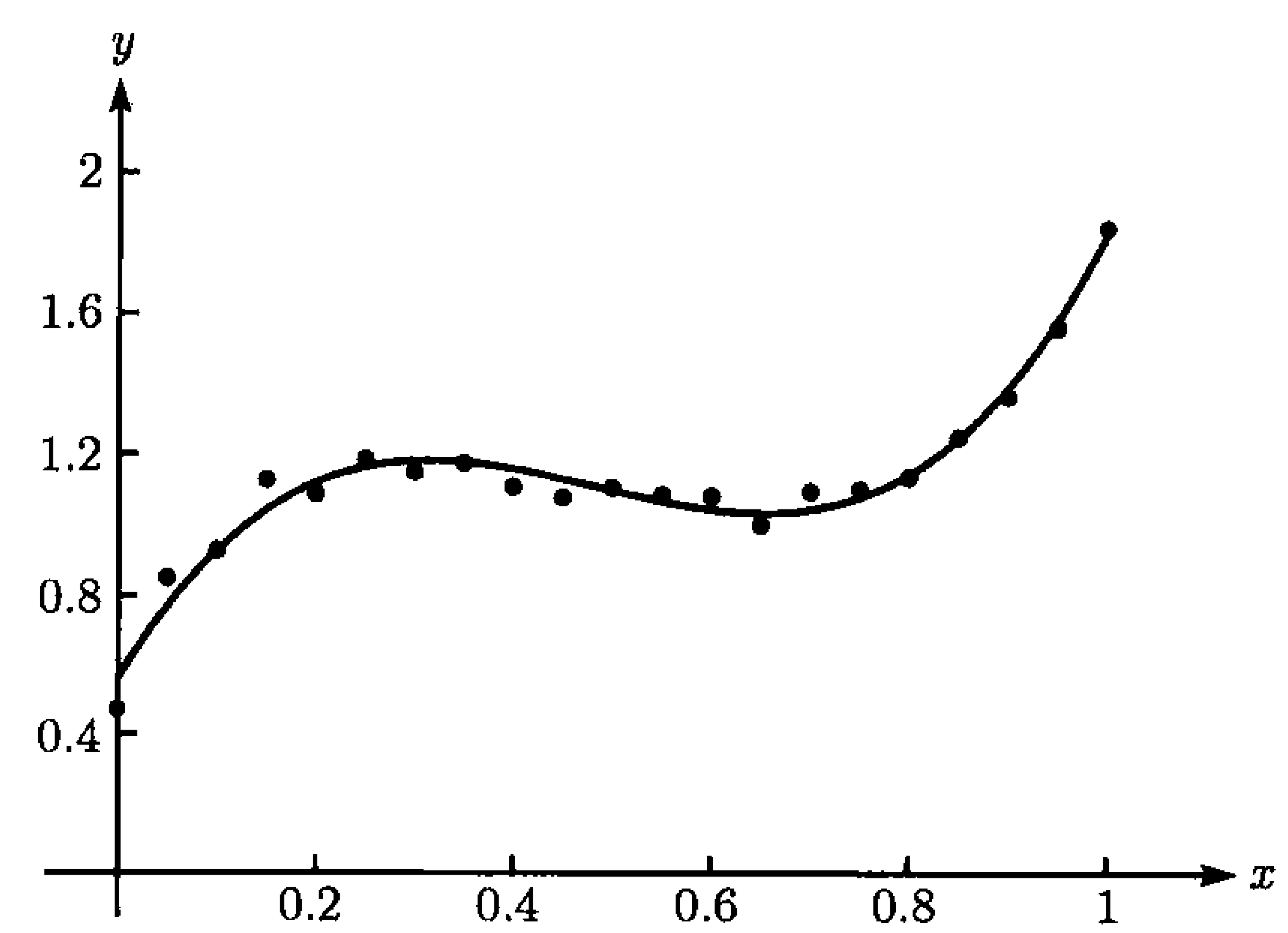


图 7-4　表 7-2 的三次多项式最小二乘拟合

MATLAB 多项式数据拟合函数是`polyfit`. 给定同样长度的向量`x`和向量`y`, `x`的分量各不相同, 调用`polyfit(x,y,n)`得到次数小于或等于 n 的最小二乘多项式拟

合. 例如, 我们将例 7.1.3 中的数据点存放在向量xx和向量yy中. 则由下列 MATLAB 指令产生图 7-5:

```
p1 = polyfit(xx,yy,1);
p2 = polyfit(xx,yy,2);
p3 = polyfit(xx,yy,3);
plot(xx,yy,'ko',xx,polyval(p1,xx),'--',xx,...
   polyval(p2,xx),'-.',xx,polyval(p3,xx),'LineWidth',2)
legend('data points','degree 1 fit',...
   'degree 2 fit','degree 3 fit')
```

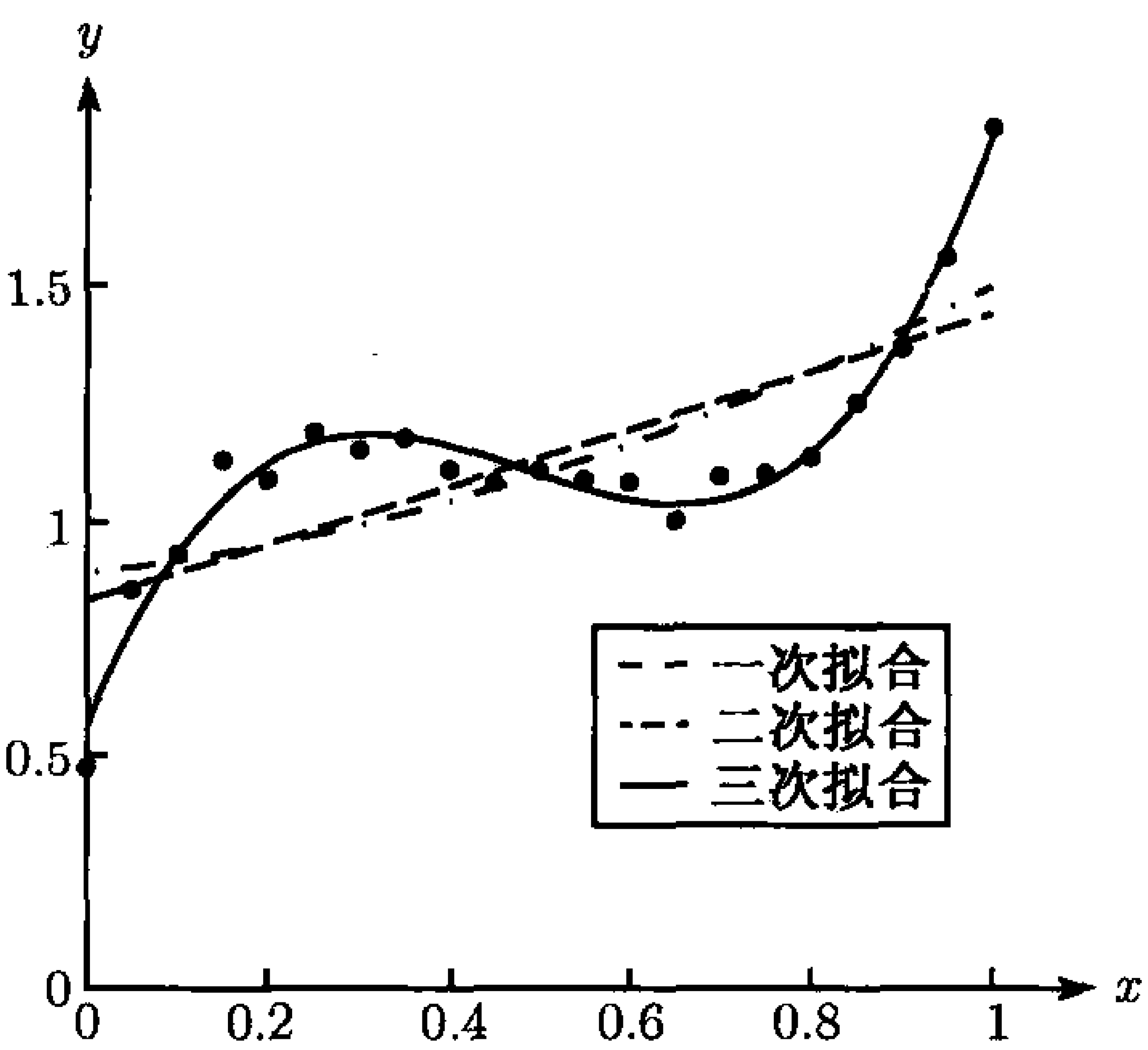


图 7-5 表 7-2 的最小二乘拟合

习题

1. 计算下列数据的线性最小二乘拟合. 画出数据和最小二乘拟合的图形. 再求最小二乘拟合的均方根误差.

x_i	y_i	x_i	y_i	x_i	y_i
0	−1.466	1.2	1.068	2.4	4.148
0.3	−0.062	1.5	1.944	2.7	4.464
0.6	0.492	1.8	2.583	3.0	5.185
0.9	0.822	2.1	3.239		

2. 用下列数据重复习题 1 中的工作:

x_i	y_i	x_i	y_i	x_i	y_i
−1.0	1.032	−0.3	1.139	0.4	−0.415
−0.9	1.563	−0.2	0.646	0.5	−0.112
−0.8	1.614	−0.1	0.474	0.6	−0.817
−0.7	1.377	0	0.418	0.7	−0.234
−0.6	1.179	0.1	0.067	0.8	−0.623
−0.5	1.189	0.2	0.371	0.9	−0.536
−0.4	0.910	0.3	0.183	1.0	−1.173

3. 对下列数据作二次多项式最小二乘拟合. 使用标准形式

$$\hat{f}(x) = a_1 + a_2 x + a_3 x^2$$

和方程组 (7.12) 的解 a_1, a_2, a_3, 再计算均方根误差, 并作数据图以及最小二乘拟合的图形.

x_i	y_i	x_i	y_i	x_i	y_i
−1.0	7.904	−0.3	0.335	0.4	−0.711
−0.9	7.452	−0.2	−0.271	0.5	0.224
−0.8	5.827	−0.1	−0.963	0.6	0.689
−0.7	4.400	0	−0.847	0.7	0.861
−0.6	2.908	0.1	−1.278	0.8	1.358
−0.5	2.144	0.2	−1.335	0.9	2.613
−0.4	0.581	0.3	−0.656	1.0	4.599

4. 本节中描述的过程称为关于离散数据的最小二乘逼近方法. 对连续函数给出的近似数据有一种相应的方法, 尽管这样做的原因与离散情况不同. 为简单起见, 考虑在区间 $0 \leqslant x \leqslant 1$ 上用二次多项式

$$\hat{f}(x) = a_1 + a_2 x + a_3 x^2$$

逼近 $f(x)$. 选择 a_1, a_2, a_3 使均方根误差

$$E = \left\{ \int_0^1 [a_1 + a_2 x + a_3 x^2 - f(x)]^2 \mathrm{d}x \right\}^{1/2}$$

达到极小. 导出最优选择 a_1, a_2, a_3 满足的线性方程组. 得到的方程组的系数矩阵与 6.5 节的希尔伯特矩阵有什么关系?

5. 本题研究从统计的观点来导出的线性最小二乘逼近估计. 设 (x_j, y_j), $1 \leqslant j \leqslant n$, 是给定的数据点. 我们感兴趣的是线性曲线拟合. 定义平均值

$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j, \quad \bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$$

自然地, 我们要求点 $(\bar{x}, \bar{y})$ 位于所求直线上. 因此, 线性函数具有形式

$$y = \bar{y} + a(x - \bar{x})$$

为确定 a, 我们极小化均方根误差

$$\sqrt{\frac{1}{n}\sum_{j=1}^{n} [y_j - (\bar{y} + a(x_j - \bar{x}))]^2}$$

即选择 a 使二次函数

$$\sum_{j=1}^{n} [y_j - (\bar{y} + a(x_j - \bar{x}))]^2 = \sum_{j=1}^{n} (y_j - \bar{y})^2 - 2a \sum_{j=1}^{n} (x_j - \bar{x})(y_j - \bar{y})$$

$$+ a^2 \sum_{j=1}^{n} (x_j - \bar{x})^2$$

达到极小. 容易得到,

$$a = \frac{\sum_{j=1}^{n} (x_j - \bar{x})(y_j - \bar{y})}{\sum_{j=1}^{n} (x_j - \bar{x})^2}$$

证明以这种方式确定的线性函数与由线性方程组 (7.6) 确定 b 和 m 得到的函数 $\hat{f}^*(x) = mx + b$ 是恒等的.

6. 设 (x_j, y_j) 是 n 个点, $x_j \geqslant 0$, $1 \leqslant j \leqslant n$. 已知 $\mu > 0$, 证明: 对于最小二乘幂函数曲线 $y = ax^{\mu}$, 有

$$a = \frac{\sum_{j=1}^{n} x_j^{\mu} y_j}{\sum_{j=1}^{n} x_j^2 \mu}$$

7. 对于 $n = 2$ 和 $n = 3$, 验证: 除了 $x_1 = \cdots = x_n =$ 常量之外, 不等式 (7.7) 成立.

8. 使用 MATLAB 内置函数`cond(A,inf)`研究方程组 (7.15) 的条件. 设 $n = 11$, $x_j = (j-1)h$, $1 \leqslant j \leqslant n$, $h = 1/(n-1)$. 对 $m = 1, 2, 3, 4, 5$, 计算该方程组的条件数. 观察条件数如何随着 m 快速递增.

9. 模仿方程组 (7.11) 的推导, 导出线性方程组 (7.13).

10. 用

$$\hat{f}(x) = a + b\mathrm{e}^{-x}$$

取代 $\hat{f}(x) = mx + b$ 的假设. 使均方根误差

$$E = \left[\frac{1}{n} \sum_{j=1}^{n} (a + b\mathrm{e}^{-x_j} - y_j)^2 \right]^{1/2}$$

达到极小化, 求 a 和 b. 并求出最优选择 a 和 b 满足的线性方程组.

提示: 为了简化工作, 先极小化 E 中的和, 然后将 (7.4) 式推到 (7.6) 式.

11. 一些更困难的曲线拟合问题可以转化为线性最小二乘逼近问题. 例如拟合函数 $y = ax^{\mu}$, 要确定 a 和 μ. 假设 (x_j, y_j), $x_j > 0$, $y_j > 0$, $1 \leqslant j \leqslant n$, 是已知点. 应用自然对数将拟合函数改写成

$$\ln y = \ln a + \mu \ln x$$

说明如何确定 a 和 μ.

12. 对于拟合函数

$$y = \frac{1}{(ax + b)^3}$$

类似于习题 11 将问题转化为一个线性最小二乘逼近问题.

7.2 本征值问题

对方阵 $\boldsymbol{A}$, 我们定义 $\boldsymbol{A}$ 的本征值概念如下. 若

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{7.21}$$

且 $\boldsymbol{v} \neq \mathbf{0}$, 则数 λ 称为 $\boldsymbol{A}$ 的本征值, 列向量 $\boldsymbol{v}$ 称为相应的本征向量. 可用矩阵 $\boldsymbol{A}$ 的本征值–本征向量对 $\{\lambda, \boldsymbol{v}\}$ 解释当 $\boldsymbol{A}$ 乘以一般的列向量 $\boldsymbol{x}$ 以及求解线性方程组 $\boldsymbol{Ax} = \boldsymbol{b}$ 时发生的情况. 本征值和本征向量也出现在许多物理问题中, 如在绷紧的弦或音乐仪器振动的固有频率中那样, 常常导致某个具有内在物理意义的 "基本解". 本节中, 我们简要地介绍本征值和本征向量的理论, 然后给出一个计算矩阵按绝对值最大的本征值的数值方法.

例 7.2.1　矩阵

$$\boldsymbol{A} = \begin{bmatrix} 1.25 & 0.75 \\ 0.75 & 1.25 \end{bmatrix} \tag{7.22}$$

的本征值–本征向量对是

$$\begin{aligned} \lambda_1 = 2, \quad \boldsymbol{v}^{(1)} &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \lambda_2 = 0.5, \quad \boldsymbol{v}^{(2)} &= \begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{aligned} \tag{7.23}$$

读者可以验证这两个本征值–本征向量对都满足 (7.21). 现在利用本征值和本征向量进一步解释 $\boldsymbol{A}$ 乘以一般的列向量 $\boldsymbol{x}$ 的含义. 对于 $\boldsymbol{x} = [x_1, x_2]^{\mathrm{T}}$, 可以直接验证

$$\boldsymbol{x} = c_1\boldsymbol{v}^{(1)} + c_2\boldsymbol{v}^{(2)} \tag{7.24}$$

其中

$$c_1 = \frac{x_1 + x_2}{2}, \quad c_2 = \frac{x_2 - x_1}{2}$$

把 $\boldsymbol{x}$ 看作 x_1x_2 平面内从原点到坐标点 (x_1, x_2) 的向量. 则 (7.24) 式表示可以把 $\boldsymbol{x}$ 写成两个向量 $c_1\boldsymbol{v}^{(1)}$ 和 $c_2\boldsymbol{v}^{(2)}$ 的和, 如图 7-6a 所示. 现在计算 $\boldsymbol{Ax}$:

$$\begin{aligned} \boldsymbol{Ax} &= \boldsymbol{A}[c_1\boldsymbol{v}^{(1)} + c_2\boldsymbol{v}^{(2)}] \\ &= c_1\boldsymbol{Av}^{(1)} + c_2\boldsymbol{Av}^{(2)} \\ &= c_1\lambda_1\boldsymbol{v}^{(1)} + c_2\lambda_2\boldsymbol{v}^{(2)} \\ \boldsymbol{Ax} &= 2c_1\boldsymbol{v}^{(1)} + 0.5c_2\boldsymbol{v}^{(2)} \end{aligned}$$

因此, 将 $\boldsymbol{x}$ 的一个分量 $c_1\boldsymbol{v}^{(1)}$ 的 2 倍和另一个分量 $c_2\boldsymbol{v}^{(2)}$ 的 0.5 倍相加就得到 $\boldsymbol{Ax}$, 如图 7-6b 所示. 图形表明计算 $\boldsymbol{Ax}$ 的问题在以本征向量 $\boldsymbol{v}^{(1)}$ 和 $\boldsymbol{v}^{(2)}$ 为坐标轴的坐标系中可以更好地表述. 这些想法可以推广到其他 2 阶方阵和其他更高阶的方阵. ■

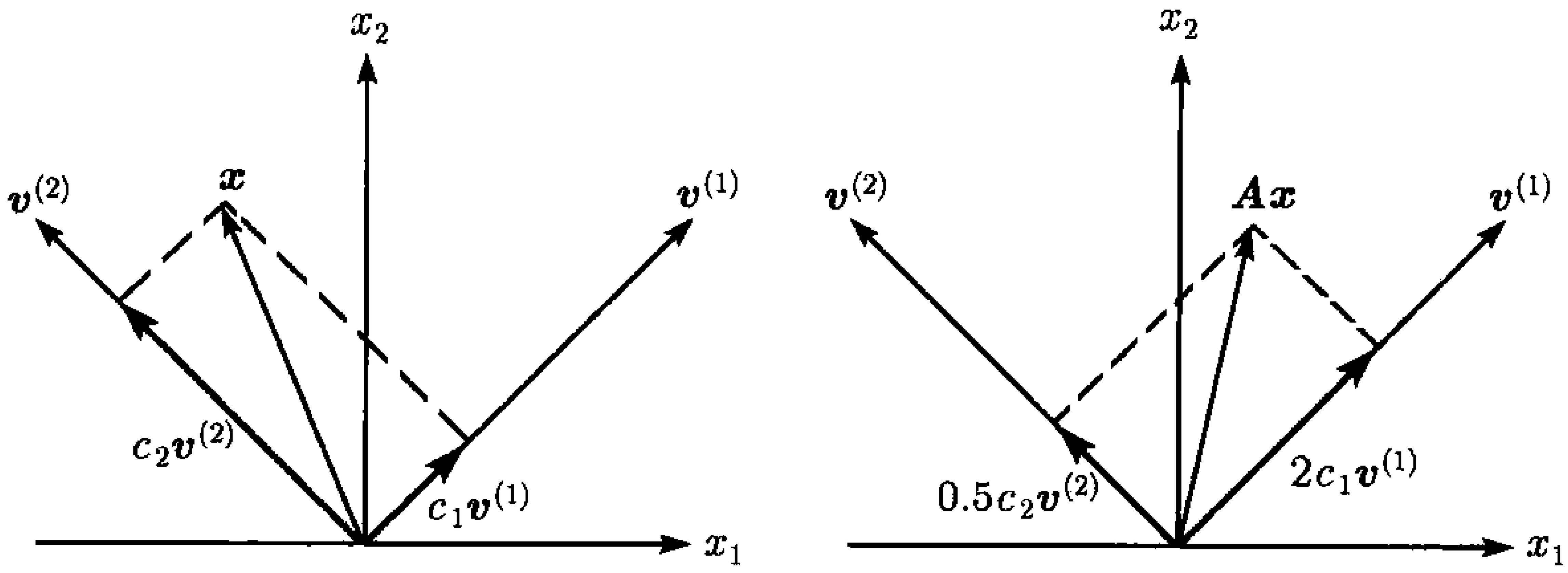


图 7-6 (a) $\boldsymbol{x} = c_1\boldsymbol{v}^{(1)} + c_2\boldsymbol{v}^{(2)}$, (b) $\boldsymbol{Ax} = 2c_1\boldsymbol{v}^{(1)} + 0.5c_2\boldsymbol{v}^{(2)}$

7.2.1 特征多项式

改写定义 (7.21) 为

$$(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{v} = 0, \quad \boldsymbol{v} \neq \boldsymbol{0} \tag{7.25}$$

这是具有系数矩阵 $\lambda\boldsymbol{I} - \boldsymbol{A}$ 和非零解 $\boldsymbol{v}$ 的齐次线性方程组. 由 6.3 节定理 6.2.10 知, 方程组具有非零解当且仅当

$$\det(\lambda \boldsymbol{I} - \boldsymbol{A}) = 0$$

函数

$$f(\lambda) = \det(\lambda \boldsymbol{I} - \boldsymbol{A})$$

称为 $\boldsymbol{A}$ 的特征多项式, 它的根是 $\boldsymbol{A}$ 的本征值. 假设 $\boldsymbol{A}$ 为 n 阶方阵, 利用行列式性质可以证明

$$f(\lambda) = \lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots + \alpha_1\lambda + \alpha_0 \tag{7.26}$$

是 n 次多项式. 对 $n = 2$ 的情况,

$$\begin{aligned} f(\lambda) &= \det \begin{bmatrix} \lambda - a_{11} & -a_{12} \\ -a_{21} & \lambda - a_{22} \end{bmatrix} \\ &= (\lambda - a_{11})(\lambda - a_{22}) - a_{21}a_{12} \\ &= \lambda^2 - (a_{11} + a_{22})\lambda + a_{11}a_{22} - a_{21}a_{12} \end{aligned}$$

公式 (7.26) 说明 n 阶矩阵 $\boldsymbol{A}$ 最多能有 n 个不同的本征值.

例 7.2.2 对于 (7.22) 中的 $\boldsymbol{A}$,

$$\begin{aligned} f(\lambda) &= \det \begin{bmatrix} \lambda - 1.25 & -0.75 \\ -0.75 & \lambda - 1.25 \end{bmatrix} \\ &= \lambda^2 - 2.5\lambda + 1 \end{aligned}$$

这个多项式的根为 $\lambda = 0.5, 2$, 正如 (7.23) 中给出的那样. ■

一旦知道本征值, 用实际的本征值替代 λ, 就可以通过求解线性方程组 (7.25) 得到本征向量. 注意到解 $\boldsymbol{v}$ 不唯一, 因为它的任何常数倍也是解. 由于有的地方舍入误差会迅速地传播及扩张, 因此 $\boldsymbol{v}$ 的求解过程并不总是简单的. 我们用一个例子说明一般的求解过程, 这里舍入误差不是一个问题.

例 7.2.3 设

$$\boldsymbol{A}=\begin{bmatrix}-7 & 13 & -16\\ 13 & -10 & 13\\ -16 & 13 & -7\end{bmatrix} \tag{7.27}$$

则

$$\begin{aligned}f(\lambda)&=\det\begin{bmatrix}\lambda+7 & -13 & 16\\ -13 & \lambda+10 & -13\\ 16 & -13 & \lambda+7\end{bmatrix}\\ &=\lambda^3+24\lambda^2-405\lambda+972\end{aligned}$$

的根为

$$\lambda_1=-36,\quad \lambda_2=9,\quad \lambda_3=3 \tag{7.28}$$

对 $\lambda=\lambda_1=-36$, 我们来求相应的本征向量 $\boldsymbol{v}=\boldsymbol{v}^{(1)}$. 如果将 $\lambda=\lambda_1$ 代入 (7.25) 中, 则 $\boldsymbol{v}$ 满足

$$(-36\boldsymbol{I}-\boldsymbol{A})\boldsymbol{v}=\boldsymbol{0}$$

$$\begin{bmatrix}-29 & -13 & 16\\ -13 & -26 & -13\\ 16 & -13 & -29\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ v_3\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\end{bmatrix}$$

如果在这个方程组中 $v_1=0$, 则可以直接证明 $v_2=v_3=0$. 因此, 对于 $\boldsymbol{v}\neq\boldsymbol{0}$, 有 $v_1\neq 0$. 由于 $\boldsymbol{v}$ 的任何倍也是解, 因此, 我们可以选择 $v_1=1$. 这就导致方程组

$$\begin{cases}-13v_2+16v_3=29\\ -26v_2-13v_3=13\\ -13v_2-29v_3=-16\end{cases}$$

方程组的解为 $v_2=-1$, $v_3=1$. 因此,

$$\boldsymbol{v}^{(1)}=[1,-1,1]^{\mathrm{T}} \tag{7.29}$$

或为这个向量的任何非零倍. 向量 $\boldsymbol{v}^{(2)}$ 和 $\boldsymbol{v}^{(3)}$ 的构造留作习题 2. ■

一般地, 并不通过求特征多项式的根来计算本征值. 本节后面, 我们讨论计算主本征值 (按绝对值最大的本征值) 的幂法. 更有效的技巧是通常先将 $\boldsymbol{A}$ 化为具有同样本征值并且更容易计算的矩阵. 不过这些方法的讨论已超出本书的范围.

7.2.2 对称矩阵的本征值

我们称矩阵 $\boldsymbol{A}=[a_{ij}]$ 是对称的, 如果

$$\boldsymbol{A}^{\mathrm{T}}=\boldsymbol{A}$$

或等价地,

$$a_{ij}=a_{ji},\quad 1\leqslant i,j\leqslant n$$

这意味着 $\boldsymbol{A}$ 直观上关于 $\boldsymbol{A}$ 的对角元素 a_{11} 和 a_{nn} 之间连线对称. 矩阵

$$\boldsymbol{A}=\begin{bmatrix} a & b & c\\ b & d & e\\ c & e & f\end{bmatrix}$$

是一般的三阶对称矩阵. 在应用中通常出现对称矩阵, 而在这种情况, 本征值问题的理论和数值分析更为简单. 对称矩阵有下列重要结论.

定理 7.2.4 设 $\boldsymbol{A}$ 是 n 阶对称矩阵. 则存在 n 个本征值–本征向量对 $\left\{\lambda_i,\boldsymbol{v}^{(i)}\right\}$, $1\leqslant i\leqslant n$, 满足下列性质:

i. 数 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 都是 $\boldsymbol{A}$ 的特征多项式 $f(\lambda)$ 的根, 依据它们的重数重复计算. 并且, 所有 λ_i 都是实数.

ii. 如果把列矩阵 $\boldsymbol{v}^{(i)}$, $1\leqslant i\leqslant n$, 看作 n 维空间的向量, 则它们相互垂直且长度为 1.

iii. 对于每个列向量 $\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$, 存在唯一的常数 $c_1,\cdots,c_n$ 满足

$$\boldsymbol{x}=c_1\boldsymbol{v}^{(1)}+\cdots+c_n\boldsymbol{v}^{(n)} \tag{7.30}$$

这些常数为

$$c_i=\sum_{j=1}^{n}x_jv_j^{(i)}=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{v}^{(i)},\quad 1\leqslant i\leqslant n \tag{7.31}$$

其中 $\boldsymbol{v}^{(i)}=\left[v_1^{(i)},\cdots,v_n^{(i)}\right]^{\mathrm{T}}$.

iv. 由

$$\boldsymbol{U}=[\boldsymbol{v}^{(1)},\boldsymbol{v}^{(2)},\cdots,\boldsymbol{v}^{(n)}] \tag{7.32}$$

定义 n 阶矩阵 $\boldsymbol{U}$. 则

$$\boldsymbol{U}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{U}=\boldsymbol{D}\equiv\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & \lambda_n\end{bmatrix} \tag{7.33}$$

并且

$$UU^{\mathrm{T}} = U^{\mathrm{T}}U = I \tag{7.34}$$

由于这个定理的证明较长并且需要用到本章没有讨论过的线性代数的概念, 因此, 我们略去定理证明. 满足 (7.34) 的矩阵称为**正交矩阵**, 在矩阵代数问题的数值分析中它们是一类重要的矩阵. 性质 (7.34) 表明向量 $\{\boldsymbol{v}^{(1)}, \cdots, \boldsymbol{v}^{(n)}\}$ 构成长度为 n 的所有列向量的**正交基**. 这种正交矩阵的例子见习题 15 和 6.2 节的习题 7.

例 7.2.5 回顾 (7.22) 中的矩阵 $\boldsymbol{A}$ 以及 (7.23) 中它的本征值–本征向量对. 图 7-6 说明本征向量是垂直的. 为了使它们长度为 1, 可用

$$\lambda_1 = 2, \quad \boldsymbol{v}^{(1)} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$\lambda_2 = 0.5, \quad \boldsymbol{v}^{(2)} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

取代 (7.23). 与 (7.32) 相应的矩阵 $\boldsymbol{U}$ 为

$$\boldsymbol{U} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

显然, $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{I}$, 而且

$$\begin{aligned} \boldsymbol{U}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{U} &= \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1.25 & 0.75 \\ 0.75 & 1.25 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix} \end{aligned}$$

正如 (7.33) 中指出的那样. ■

我们常将结论 (7.33) 写成

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$$

$\boldsymbol{A}$ 的这种分解在数学和别的学科的许多应用中非常有用.

7.2.3 非对称本征值问题

对于非对称矩阵, 本征值和本征向量有许多可能情况. 特征多项式 $f(\lambda)$ 的根可能是复数, 这些根导致分量为复数的本征向量. 如果 $f(\lambda)$ 有重根, 则有可能没法把任意列向量 $\boldsymbol{x}$ 写成本征向量的组合; 即使能写成 (7.30) 那样的表达式, 系数也不

再有如 (7.31) 那样的简单公式. 非对称本征值问题很重要, 但在这里讨论它的一般理论过于复杂.

例 7.2.6 我们来看一个本征值为复数的矩阵. 设

$$\boldsymbol{A}=\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

它的特征多项式是

$$f(\lambda)=\det\begin{bmatrix} \lambda & -1 \\ 1 & \lambda \end{bmatrix}=\lambda^2+1$$

$f(\lambda)$ 的根是复数

$$\lambda_1=i\equiv\sqrt{-1},\quad \lambda_2=-i$$

并且相应本征向量是

$$\boldsymbol{v}^{(1)}=\begin{bmatrix} i \\ -1 \end{bmatrix},\quad \boldsymbol{v}^{(2)}=\begin{bmatrix} i \\ 1 \end{bmatrix}$$ ■

由 n 阶对称矩阵的定理 7.2.4 知道, 存在使本征向量正交的 n 个本征值-本征向量对. 对于非对称矩阵, 下例说明, 重本征值相应的独立本征向量个数可能较少.

例 7.2.7 考虑矩阵

$$\boldsymbol{A}=\begin{bmatrix} a & 1 & 0 \\ 0 & a & 1 \\ 0 & 0 & a \end{bmatrix}$$

其中 a 是常数. 留给读者一个习题, 验证 $\lambda=a$ 是 $\boldsymbol{A}$ 的 3 重本征值, 并且对某些 $c\neq 0$, 任何相应的本征向量必须具有下列形式

$$\boldsymbol{v}=c\,[1,0,0]^{\mathrm{T}}$$

因此, 对于三个相等的本征值 $\lambda_1=\lambda_2=\lambda_3=a$, 若不计非零的乘数, 我们只有一个本征向量

$$\boldsymbol{v}=[1,0,0]^{\mathrm{T}}$$ ■

7.2.4　幂法

现在描述计算 $\boldsymbol{A}$ 的绝对值最大的本征值的数值方法. 作为一般的计算过程, 它有一些限制, 但它的计算程序简单, 并且对许多矩阵它是一个令人满意的方法. 矩阵 $\boldsymbol{A}$ 不必对称, 但要求 $\boldsymbol{A}$ 的本征值必须满足

$$|\lambda_1| > |\lambda_2| \geqslant \cdots \geqslant |\lambda_n| \tag{7.35}$$

其中这些本征值依据它们作为特征多项式的根的重数重复列出.

我们定义计算 λ_1 及其相应的本征向量 $\boldsymbol{v}^{(1)}$ 的迭代过程. 选择 $\boldsymbol{v}^{(1)}$ 的初始猜测, 称它为 $\boldsymbol{z}^{(0)}$. 一般地, 它是随机选择的, 通常选择由随机数生成各分量. 定义

$$\boldsymbol{w}^{(1)} = \boldsymbol{A}\boldsymbol{z}^{(0)}$$

并设 α_1 是 $\boldsymbol{w}^{(1)}$ 按绝对值最大的分量. 如果这样的分量多于一个, 则选择第一个这样的分量为 α_1. 然后定义

$$\boldsymbol{z}^{(1)} = \frac{1}{\alpha_1}\boldsymbol{w}^{(1)}$$

重复这个迭代过程. 定义

$$\boldsymbol{w}^{(m)} = \boldsymbol{A}\boldsymbol{z}^{(m-1)} \tag{7.36}$$

并设 α_m 是 $\boldsymbol{w}^{(m)}$ 按绝对值最大的分量. 定义

$$\boldsymbol{z}^{(m)} = \frac{1}{\alpha_m}\boldsymbol{w}^{(m)} \tag{7.37}$$

则, 粗略地讲, 向量 $\boldsymbol{z}^{(m)}$ 将收敛于 $\boldsymbol{v}^{(1)}$ 的某个倍数.

用这个过程求 λ_1, 可选取向量 $\boldsymbol{z}^{(m)}$ 和 $\boldsymbol{w}^{(m)}$ 的某个非零分量, 比如为第 k 个, 并固定 k. 如果 l 充分大, 则可选 $\boldsymbol{z}^{(l)}$ 按绝对值最大的分量作为这第 k 个分量. 定义

$$\lambda_1^{(m)} = \frac{w_k^{(m)}}{z_k^{(m-1)}} \tag{7.38}$$

其中 $z_k^{(m-1)}$ 表示 $\boldsymbol{z}^{(m-1)}$ 的第 k 个分量. 可以证明, 当 $m \to \infty$ 时, $\lambda_1^{(m)}$ 收敛于 λ_1.

例 7.2.8　对于前面例子的 (7.22) 应用幂法, 其中

$$\boldsymbol{A} = \begin{bmatrix} 1.25 & 0.75 \\ 0.75 & 1.25 \end{bmatrix} \tag{7.39}$$

结果如表 7-3 所示. 其中相继两次的差 $\lambda_1^{(m)} - \lambda_1^{(m-1)}$ 和它们相继两次的比的列表明, 存在一种有规则的模式收敛. 其原因将在下面解释. 注意在这个例子中, $\lambda_1 = 2$, $\boldsymbol{v}^{(1)} = [1,1]^{\mathrm{T}}$, 并且数值结果收敛到这些值.

表 7-3 (7.39) 的幂法举例

m	$z_1^{(m)}$	$z_2^{(m)}$	$\lambda_1^{(m)}$	$\lambda_1^{(m)}-\lambda_1^{(m-1)}$	比率
0	1.0	0.5			
1	1.0	0.846 153 8	1.625 000 0		
2	1.0	0.959 183 7	1.884 615 4	2.596E−1	
3	1.0	0.989 637 3	1.969 387 8	8.477E−2	0.327
4	1.0	0.997 399 2	1.992 228 0	2.284E−2	0.269
5	1.0	0.999 349 2	1.998 049 4	5.821E−3	0.255
6	1.0	0.999 837 3	1.999 511 9	1.462E−3	0.251
7	1.0	0.999 959 3	1.999 877 9	3.660E−4	0.250

7.2.5 幂法的收敛性

为了分析幂法的收敛性, 我们仅考虑 $\boldsymbol{A}$ 是对称矩阵的情况. 并且假设 λ_1 如同 (7.35) 中那样是主本征值. 我们先证明

$$\boldsymbol{z}^{(m)}=\sigma_m\frac{\boldsymbol{A}^m\boldsymbol{z}^{(0)}}{\|\boldsymbol{A}^m\boldsymbol{z}^{(0)}\|} \tag{7.40}$$

其中 $\sigma_m=\pm1$. 对于 $m=1$, 因为 α_1 是 $\boldsymbol{w}^{(1)}$ 按绝对值最大的分量, 所以

$$\alpha_1=\pm\left\|\boldsymbol{w}^{(1)}\right\|=\pm\left\|\boldsymbol{A}\boldsymbol{z}^{(0)}\right\|$$

于是

$$\boldsymbol{z}^{(1)}=\frac{1}{\alpha_1}\boldsymbol{w}^{(1)}=\pm\frac{\boldsymbol{A}\boldsymbol{z}^{(0)}}{\left\|\boldsymbol{A}\boldsymbol{z}^{(0)}\right\|}$$

$\boldsymbol{z}^{(1)}$ 的绝对值最大的分量, 或几个这样的最大分量中的第一个分量, 就等于 1. 因此, 右边符号依据最大分量要求来选择. 使用与 $m=1$ 情况一样的思想方法, 可以用归纳法证明一般情况 (7.40). 选择符号 σ_m 使 (7.40) 右边的最大分量为 1, 因为由 $\boldsymbol{z}^{(m)}$ 的定义 (7.37) 这个结论成立.

利用 (7.30), 记

$$\boldsymbol{z}^{(0)}=c_1\boldsymbol{v}^{(1)}+\cdots+c_n\boldsymbol{v}^{(n)} \tag{7.41}$$

其中 c_j 由 (7.31) 给出并用 $\boldsymbol{z}^{(0)}$ 取代 $\boldsymbol{x}$. 我们假设 $c_1\neq0$. 如果随机选择 $\boldsymbol{z}^{(0)}$, 则极有可能 $c_1\neq0$; 即使 $c_1=0$, 由于舍入误差, 幂法通常会引进一个修正的 $c_1\neq0$. 在 (7.41) 中用 $\boldsymbol{A}$ 乘以 $\boldsymbol{z}^{(0)}$ 得到

$$\begin{aligned}\boldsymbol{A}\boldsymbol{z}^{(0)}&=c_1\boldsymbol{A}\boldsymbol{v}^{(1)}+\cdots+c_n\boldsymbol{A}\boldsymbol{v}^{(n)}\\&=\lambda_1c_1\boldsymbol{v}^{(1)}+\cdots+\lambda_nc_n\boldsymbol{v}^{(n)}\end{aligned}$$

重复用 $\boldsymbol{A}$ 相乘得到

$$\begin{aligned}\boldsymbol{A}^m\boldsymbol{z}^{(0)} &= \lambda_1^m c_1\boldsymbol{v}^{(1)}+\cdots+\lambda_n^m c_n\boldsymbol{v}^{(n)}\\ &= \lambda_1^m\left[c_1\boldsymbol{v}^{(1)}+\left(\frac{\lambda_2}{\lambda_1}\right)^m c_2\boldsymbol{v}^{(2)}+\cdots+\left(\frac{\lambda_n}{\lambda_1}\right)^m c_n\boldsymbol{v}^{(n)}\right]\end{aligned}$$

由 (7.40) 知,

$$\boldsymbol{z}^{(m)}=\sigma_m\left(\frac{\lambda_1}{|\lambda_1|}\right)^m\cdot\frac{c_1\boldsymbol{v}^{(1)}+(\frac{\lambda_2}{\lambda_1})^m c_2\boldsymbol{v}^{(2)}+\cdots+(\frac{\lambda_n}{\lambda_1})^m c_n\boldsymbol{v}^{(n)}}{\|c_1\boldsymbol{v}^{(1)}+(\frac{\lambda_2}{\lambda_1})^m c_2\boldsymbol{v}^{(2)}+\cdots+(\frac{\lambda_n}{\lambda_1})^m c_n\boldsymbol{v}^{(n)}\|} \tag{7.42}$$

当 $m\to\infty$ 时, 对 $2\leqslant j\leqslant n$, 项 $(\lambda_j/\lambda_1)^m\to 0$, 且 $(\lambda_2/\lambda_1)^m$ 的绝对值最大. 而且,

$$\sigma_m\left(\frac{\lambda_1}{|\lambda_1|}\right)^m=\pm 1$$

因此, 当 $m\to\infty$ 时, (7.42) 中大多数项收敛于 0, 并且我们可以从分子和分母中消去 c_1, 从而 $\boldsymbol{z}^{(m)}$ 看上去像

$$\boldsymbol{z}^{(m)}\approx\hat{\sigma}_m\frac{\boldsymbol{v}^{(1)}}{\|\boldsymbol{v}^{(1)}\|}$$

其中 $\hat{\sigma}_m=\pm 1$. 如果将 $\boldsymbol{z}^{(m)}$ 正规化, 总有某个特别的分量是正的, 则

$$\boldsymbol{z}^{(m)}\to\pm\frac{\boldsymbol{v}^{(1)}}{\|\boldsymbol{v}^{(1)}\|}\equiv\hat{\boldsymbol{v}}^{(1)} \tag{7.43}$$

具有一个与 m 无关的固定符号. 通常用 α_m 相除使前面的符号正规化, 但并不总是这样, 见习题 18. 对某个常数 $c>0$, $\boldsymbol{z}^{(m)}$ 中误差满足

$$\left\|\boldsymbol{z}^{(m)}-\hat{\boldsymbol{v}}^{(1)}\right\|\leqslant c\left|\frac{\lambda_2}{\lambda_1}\right|^m,\quad m\geqslant 0 \tag{7.44}$$

类似于上面的分析可以知道, $\lambda_1^{(m)}$ 收敛于 λ_1, 并具有同类的误差界. 而且, 如果还假设

$$|\lambda_1|>|\lambda_2|>|\lambda_3|\geqslant|\lambda_4|\geqslant\cdots\geqslant|\lambda_n|\geqslant 0 \tag{7.45}$$

则可以证明, 当 $m\to\infty$ 时, 对某个常量 c, 有

$$\lambda_1-\lambda_1^{(m)}\approx c\left(\frac{\lambda_2}{\lambda_1}\right)^m \tag{7.46}$$

例如, 回顾表 7-2 中关于矩阵 (7.39) 的数值结果. 那里 $\lambda_1=2,\lambda_2=0.5$, 并且 $\lambda_2/\lambda_1=0.25$. 注意到表中的比率接近 0.25, 这与 (7.46) 一致. 然而, 如果 λ_2 和 λ_3 是共轭复本征值, 则假设 (7.35) 不成立; 并且, 如果 λ_2 和 λ_3 是符号相反的实数, 则 (7.35) 也不成立. 幂法十分有用, 但程序很难作为一个自动的方法.

例 7.2.9 对于对称矩阵

$$A = \begin{bmatrix} -7 & 13 & -16 \\ 13 & -10 & 13 \\ -16 & 13 & -7 \end{bmatrix} \tag{7.47}$$

应用幂法. 由前面 (7.28) 知道, 本征值是

$$\lambda_1 = -36, \quad \lambda_2 = 9, \quad \lambda_1 = 3$$

并且与 λ_1 对应的本征向量 $\boldsymbol{v}^{(1)}$ 是

$$\boldsymbol{v}^{(1)} = [1, -1, 1]^{\mathrm{T}}$$

使用幂法的结果在表 7-4 中给出. 注意到相继两个 $\lambda_1^{(m)}$ 的差的比率接近

$$\frac{\lambda_2}{\lambda_1} = -0.25$$

再注意从一个迭代到下一个迭代 $\boldsymbol{z}^{(m)}$ 最大的分量的变化. 为了说明问题, 初始猜测 $\boldsymbol{z}^{(0)}$ 选择得比较接近 $\boldsymbol{v}^{(1)}$. ■

表 7-4 (7.47) 的幂法

m	$z_1^{(m)}$	$z_2^{(m)}$	$z_3^{(m)}$	$\lambda_1^{(m)}$	$\lambda_1^{(m)} - \lambda_1^{(m)}$	比率
0	1.000 000	−0.800 000	0.900 000			
1	−0.972 477	1.000 000	−1.000 000	−31.800 00		
2	1.000 000	−0.995 388	0.993 082	−36.820 75	−5.02E+0	
3	0.998 265	−0.999 229	1.000 000	−35.829 36	9.91E−1	−0.197
4	1.000 000	−0.999 775	0.999 566	−36.040 35	−2.11E−1	−0.213
5	0.999 891	−0.999 946	1.000 000	−35.990 13	5.02E−2	−0.238
6	1.000 000	−0.999 986	0.999 973	−36.002 45	−1.23E−2	−0.245
7	0.999 993	−0.999 997	1.000 000	−35.999 39	3.06E−3	−0.249

迭代 $\left\{\lambda_1^{(m)}\right\}$ 的误差结果 (7.46) 可用来产生一种加速方法. 对于所有充分大的 m 值, 由 (7.46) 可得

$$\lambda_1 - \lambda_1^{(m+1)} \approx r(\lambda_1 - \lambda_1^{(m)}), \quad r = \lambda_2/\lambda_1 \tag{7.48}$$

利用

$$r \approx \frac{\lambda_1^{(m+1)} - \lambda_1^{(m)}}{\lambda_1^{(m)} - \lambda_1^{(m-1)}} \tag{7.49}$$

选择 r. 此式的论证留作习题 19(d). 利用这个 r, 从 (7.48) 中解出 λ_1,

$$\begin{aligned}\lambda_1 &\approx \frac{1}{1-r}[\lambda_1^{(m+1)} - r\lambda_1^{(m)}] \\ &= \lambda_1^{(m+1)} + \frac{r}{1-r}[\lambda_1^{(m+1)} - \lambda_1^{(m)}]\end{aligned} \tag{7.50}$$

此式称为**艾特肯外推公式**. 同样给出误差估计

$$\lambda_1 - \lambda_1^{(m+1)} \approx \frac{r}{1-r}[\lambda_1^{(m+1)} - \lambda_1^{(m)}] \tag{7.51}$$

它称为**艾特肯误差估计公式**. 在同样的假设 (7.45) 下, 对于迭代 $\{\boldsymbol{z}^{(m)}\}$ 可以证明类似的结论. 艾特肯外推的有关材料在 3.4 节讨论求根的不动点迭代方法时曾给出过.

例 7.2.10 表 7-4 中, 取 $m+1=7$. 则由 (7.51) 得

$$\begin{aligned}\lambda_1 - \lambda_1^{(7)} &\approx \frac{r}{1-r}\left[\lambda_1^{(7)} - \lambda_1^{(6)}\right] \\ &= \frac{-0.249}{1+0.249}[0.003\ 06] \doteq -0.000\ 61\end{aligned}$$

这是实际的误差. 而艾特肯公式 (7.50) 给出 λ_1 的具有 7 位有效数字的答案. ■

7.2.6 MATLAB 本征值计算

MATLAB 的内置函数`eig`计算本征值和相对应的本征向量. 设 $\boldsymbol{A}$ 为给定的 n 阶方阵. 则调用

```
Lambda = eig(A)
```

得到以 n 个本征值为分量的一个 n 维本征向量. 调用

```
[V,D] = eig(A)
```

得到两个 $n\times n$ 矩阵 $\boldsymbol{V}$ 和 $\boldsymbol{D}$, 其中 $\boldsymbol{D}$ 为对角阵, 其对角元素为 $\boldsymbol{A}$ 的本征值, 而 $\boldsymbol{V}$ 的列是相应的本征向量. 这些矩阵满足关系 $\boldsymbol{AV}=\boldsymbol{VD}$. 对于对称阵 $\boldsymbol{A}$, $\boldsymbol{V}$ 的列向量是正交的, 从而 $\boldsymbol{V}$ 总是可逆的. 当 $\boldsymbol{A}$ 是非对称的并有重本征值时, 矩阵 $\boldsymbol{V}$ 可能奇异或接近奇异.

MATLAB 程序 我们给出幂法的 MATLAB 程序.

```
function [lambda, eigenvec, diff, ratio] = ...
            power_method(A,x_0,index,err_tol,max_iterates)
%
% The calling sequence for power_method is
%
% [lambda, eigenvec, diff, ratio]
%          = power_method(A,x_0,index,err_tol,max_iterates)
%
% It implements the power method for finding the eigenvalue
% of A which is largest in magnitude.
%
% The vector x_0 is the user's initial guess at the
% eigenvector corresponding to the eigenvalue of
% largest magnitude.  If the user has no idea of how to
% choose x_0, then we suggest using
%    x_0 = rand(n,1)
% with n the order of A. DO NOT use x_0 = 0, the zero vector.
%
% The input parameter index is used to specify the method of
% approximating the eigenvalue.  With index=0, we generate a
% random vector, call it v, and we use
%    lambda approximately (v'*w)/(v'*z)
% with Az=w.  For index in [1,n], we use
% lambda approximately w(index)/z(index).
%
% The parameter err_tol is used in the error test.  However,
% note this test needs to be made more sophisticated.  It uses
% error approx lambda_present - lambda_previous
% at each step of the iteration, and this is not an accurate
% estimate for slowly convergent iterations.
%
% The output variables are:
% lambda:  A vector of the power method iterations for the
%          approximate eigenvalue of largest magnitude.
% eigenvec:  The corresponding normalized eigenvector.
% diff:  The differences of the successive values in lambda.
% ratio:  The ratios of the successive values in diff.
%         The values of diff and ratio are of use in
%         examining the rate of convergence of the
%         power method.

% Initialization.
% Find the order of A.
n = size(A,1);
x_init = zeros(n,1);
lambda = zeros(max_iterates,1);
```

```
% Define x_init and require it to be a column vector.
for i=1:n
   x_init(i) = x_0(i);
end

% Find the max element in size.
[max_x,i_max] = max(abs(x_init));
% Normalize the initial vector to have an
% infinity norm of 1.
z = x_init/x_init(i_max);
% If index=0, generate a special vector for the
% approximate eigenvalue computation.
if index == 0
   spec_vec = rand(1,n);
end

format long
error = inf;
it_count = 0;
% Main loop.
while abs(error) > err_tol & it_count < max_iterates
   it_count = it_count + 1;
   w = A*z;
   [max_w,i_max] = max(abs(w));
% Calculate approximate eigenvalue.
   if index == 0
      lambda(it_count) = (spec_vec*w)/(spec_vec*z);
   else
      lambda(it_count) = w(index)/z(index);
   end

   if it_count > 1
      error = lambda(it_count) - lambda(it_count-1);
   end

   z = w/w(i_max); % Save approximate eigenvector.
end

if it_count == max_iterates
   disp(' ')
   disp('*** Your answers may possibly not..
%                 satisfy your error tolerane.')
end

% Setting of output variables.
```

```
eigenvec = z;
lambda = lambda(1:it_count);
diff = zeros(it_count,1);
ratio = zeros(it_count,1);
diff(2:it_count) = lambda(2:it_count)-lambda(1:it_count-1);
ratio(3:it_count) = diff(3:it_count)./diff(2:it_count-1);
```

习题

1. 找出下列矩阵的特征多项式、本征值和本征向量.

(a) $\begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix}$ (b) $\begin{bmatrix} 2 & 36 \\ 36 & 23 \end{bmatrix}$ (c) $\begin{bmatrix} -9 & 6 \\ 3 & 8 \end{bmatrix}$

2. 对 (7.27), 求本征值 λ_2 和 λ_3 对应的本征向量 $\boldsymbol{v}^{(2)}$ 和 $\boldsymbol{v}^{(3)}$.

3. 求

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

的特征多项式、本征值和本征向量.

4. 设 $\boldsymbol{A} = \begin{bmatrix} 5 & -2 \\ -1 & 4 \end{bmatrix}$. 取 $\boldsymbol{z}^{(0)} = [1,1]^{\mathrm{T}}$, 应用幂法. 确定序列 $\lambda^{(m)}, \boldsymbol{z}^{(m)}, m \geqslant 1$. 该方法收敛于主本征值和相应的本征向量吗? 如果不是, 你建议做些什么?

5. 验证例 7.2.7 中给出的结论.

6. 考虑矩阵

$$\boldsymbol{A}_1 = \begin{bmatrix} a & 1 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

其中 a 是常数. 确定所有的本征值和相应的本征向量. 把 $\boldsymbol{A}_1$ 与例 7.2.7 中 $\boldsymbol{A}$ 的结果作比较.

7. 设 $\boldsymbol{A}$ 是 n 阶方阵, 并如 (7.26) 中那样设 $f(\lambda)$ 是 $\boldsymbol{A}$ 的特征多项式.

(a) 证明 $f(\lambda)$ 的常数项 a_0 是

$$\alpha_0 = (-1)^n \det(\boldsymbol{A})$$

(b) 根据读者掌握的行列式知识, 将 $f(\lambda)$ 按 λ 的乘幂展开, 证明 λ^{n-1} 的系数 α_{n-1} 为

$$\alpha_{n-1} = -(a_{11} + a_{22} + \cdots + a_{nn})$$

称括号中的项为 $\boldsymbol{A}$ 的迹, 它是 $\boldsymbol{A}$ 的对角元之和.

8. 继续习题 7, 证明

$$\det(\boldsymbol{A}) = \lambda_1\lambda_2\cdots\lambda_n, \quad \mathrm{trace}(\boldsymbol{A}) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$$

其中 $\lambda_1, \cdots, \lambda_n$ 是 $f(\lambda)$ 的根, 每个 λ_i 重复的次数与它的重数一样多.

提示: 展开 $f(\lambda) = (\lambda - \lambda_1)\cdots(\lambda - \lambda_n)$.

9. 设 $\boldsymbol{Ax}=\lambda\boldsymbol{x}$ 且 $\boldsymbol{x}\neq\boldsymbol{0}$. 利用矩阵范数 (6.87) 证明 $|\lambda|\leqslant\|\boldsymbol{A}\|$.

提示: 使用 (6.88) 的性质 P4. 这给出了界定矩阵 $\boldsymbol{A}$ 的全部本征值的一种简单方法.

10. 设 $\boldsymbol{A}$ 是方阵, c_0 是一个常数. 证明 λ 是 $\boldsymbol{A}$ 的本征值当且仅当 $\lambda+c_0$ 是 $\boldsymbol{A}+c_0\boldsymbol{I}$ 的本征值.

11. 通过直接计算, 证明 $n\times n$ 矩阵

$$\boldsymbol{A}=[a_1,a_2,\cdots,a_n]^{\mathrm{T}}[1,1,\cdots,1]$$

有一个本征值 $\lambda=a_1+a_2+\cdots+a_n$, 而其余的本征值全为 0.

12. 证明矩阵 (7.27) 的本征向量满足定理 7.2.4, 倘若它们被适当正规化表示成长度为 1 的向量. 例如, (7.29) 中的 $\boldsymbol{v}^{(1)}$ 化为

$$\boldsymbol{v}^{(1)}=\left[\frac{1}{\sqrt{3}},\frac{-1}{\sqrt{3}},\frac{1}{\sqrt{3}}\right]^{\mathrm{T}}$$

(向量长度是其分量的平方和之平方根. 对一般的向量

$$\boldsymbol{x}=[x_1,\cdots,x_n]^{\mathrm{T}}$$

$\boldsymbol{x}$ 的长度是 $\sqrt{x_1^2+\cdots+x_n^2}$). 给出 (7.32) 的矩阵 $\boldsymbol{U}$, 并证明它满足 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}=\boldsymbol{I}$.

13. 对习题 1(b) 中的矩阵, 重复习题 12 中的工作.

14. 计算

(a) $\begin{bmatrix}1&0\\0&1\end{bmatrix}$　　(b) $\begin{bmatrix}1&0\\1&1\end{bmatrix}$

的特征多项式、本征值和本征向量. 结果与定理 7.2.4 矛盾吗?

15. 证明: 对任何角度 θ,

$$\boldsymbol{U}=\begin{bmatrix}\cos\theta&\sin\theta\\-\sin\theta&\cos\theta\end{bmatrix}$$

是正交矩阵. 因为通过坐标轴的旋转改变直角坐标之间的关系, 所以这些矩阵称为**旋转矩阵**.

16. 对习题 1 的矩阵应用幂法. 对每种情况, 计算相继两次迭代的差 $\lambda_1^{(m)}-\lambda_1^{(m-1)}$ 和相继两次迭代的比率. 试用 (7.46) 和这些比率估计 λ_2.

17. 对下列矩阵重复习题 16 中的工作:

(a) $\begin{bmatrix}1&\frac{1}{2}\\\frac{1}{2}&\frac{1}{3}\end{bmatrix}$　　(b) $\begin{bmatrix}1&5&-8\\5&-2&5\\-8&5&1\end{bmatrix}$

(c) $\begin{bmatrix}1&2&3\\2&3&4\\3&4&5\end{bmatrix}$　　(d) $\begin{bmatrix}33&16&72\\-24&-10&-57\\-8&-4&-17\end{bmatrix}$

(e) $\begin{bmatrix}1&\frac{1}{2}&\frac{1}{3}\\\frac{1}{2}&\frac{1}{3}&\frac{1}{4}\\\frac{1}{3}&\frac{1}{4}&\frac{1}{5}\end{bmatrix}$　　(f) $\begin{bmatrix}6&4&4&1\\4&6&1&4\\4&1&6&4\\1&4&4&6\end{bmatrix}$

18. 对矩阵 (7.47) 应用幂法, 取 $z^{(0)}=[1,0,1]^{\mathrm{T}}$. 描述近似向量 $z^{(m)}$ 的收敛模式.

19. (a) 使用 $\lambda_1^{(m)}$ 的定义 (7.38) 以及假设 (7.35), 证明

$$\lim_{m\to\infty}\lambda_1^{(m)}=\lambda_1$$

(b) 证明: 对某个常量 c,

$$\left|\lambda_1-\lambda_1^{(m)}\right|\leqslant c\left|\lambda_2/\lambda_1\right|^m,\quad m\geqslant 0$$

(c) 在假设 (7.45) 下, 证明公式

$$\lambda_1-\lambda_1^{(m)}\approx c(\lambda_2/\lambda_1)^m,\quad c=\text{常量}$$

当 $m\to\infty$ 时, 逼近越来越好. 更精确地,

$$\lim_{m\to\infty}\frac{\lambda_1-\lambda_1^{(m)}}{(\lambda_2/\lambda_1)^m}=c$$

它通常是一个非零数.

(d) 使用 (7.46) 证明

$$\lim_{m\to\infty}\frac{\lambda_1^{(m+1)}-\lambda_1^{(m)}}{\lambda_1^{(m)}-\lambda_1^{(m-1)}}=\frac{\lambda_2}{\lambda_1}$$

这就证明 (7.49) 是合理的.

20. 对于对称阵 $\boldsymbol{A}$, 为了计算 λ_1, 定义下列变形的幂法:

$$\lambda_1^{(m)}=\frac{(z^{(m)})^{\mathrm{T}}\boldsymbol{A}(z^{(m)})}{(z^{(m)})^{\mathrm{T}}(z^{(m)})}=\frac{(z^{(m)})^{\mathrm{T}}(w^{(m+1)})}{(z^{(m)})^{\mathrm{T}}(z^{(m)})}$$

利用定理 7.2.4, 验证

$$\left|\lambda_1-\lambda_1^{(m)}\right|\leqslant c\left(\frac{\lambda_2}{\lambda_1}\right)^{2m},\quad m\geqslant 0$$

这样收敛较快, 误差减少的速度是 $(\lambda_2/\lambda_1)^2$ 而不是 (λ_2/λ_1). 将它应用到矩阵 (7.47) 来说明这个方法.

21. 比较

$$\boldsymbol{A}=\begin{bmatrix}101 & -90\\ 110 & -98\end{bmatrix},\quad \hat{\boldsymbol{A}}=\begin{bmatrix}100.999 & -90.001\\ 110 & -98\end{bmatrix}$$

的本征值. 你应该如何描述 $\boldsymbol{A}$ 的本征值的灵敏度?

7.3 非线性方程组

非线性方程组广泛地出现在描述现实世界各种现象的数学模型中, 它们常常比线性方程组更接近实际. 本节中, 我们对非线性方程组及求解它们的数值方法给出一个初步介绍. 为了让读者对非线性方程组可解性有一些直观认识, 并说明在它们

的数值解中使用的一些主要的思想方法，本节先考察两个非线性方程的方程组及其求解的数值方法. 然后，对任意阶的非线性方程组给出求解方法.

考虑下列两个非线性方程的方程组：

$$
\begin{aligned}
f(x,y) &\equiv x^2+4y^2-9=0 \\
g(x,y) &\equiv 18y-14x^2+45=0
\end{aligned}
\tag{7.52}
$$

怎样的一对 (x,y) 能同时满足两个方程？并且如何数值计算它们的值呢？更一般地，考虑一般的两个非线性方程的方程组：

$$
\begin{aligned}
f(x,y) &= 0 \\
g(x,y) &= 0
\end{aligned}
\tag{7.53}
$$

如何从几何的角度来阐述这些方程和它们的数值解呢？

如果在 xyz 空间内作 $z=f(x,y)$ 的图形，则 $f(x,y)=0$ 的解 (x,y) 位于 xy 平面和 $z=f(x,y)$ 图形的交集上. 我们称这个交集为 $f(x,y)$ 的“零曲线”. 例如，(7.52) 的函数 f 和 g 的零曲线在图 7-7 中给出. 为了直观地看出同时满足 (7.53) 中两个方程的点 (x,y)，我们观察 f 和 g 的零曲线的交集. 对方程组 (7.52)，图 7-7 说明了这种情况. 零曲线在 4 个点上相交，每个点对应方程组 (7.52) 的一个解. 例如，在 $(1,-1)$ 附近有一个解.

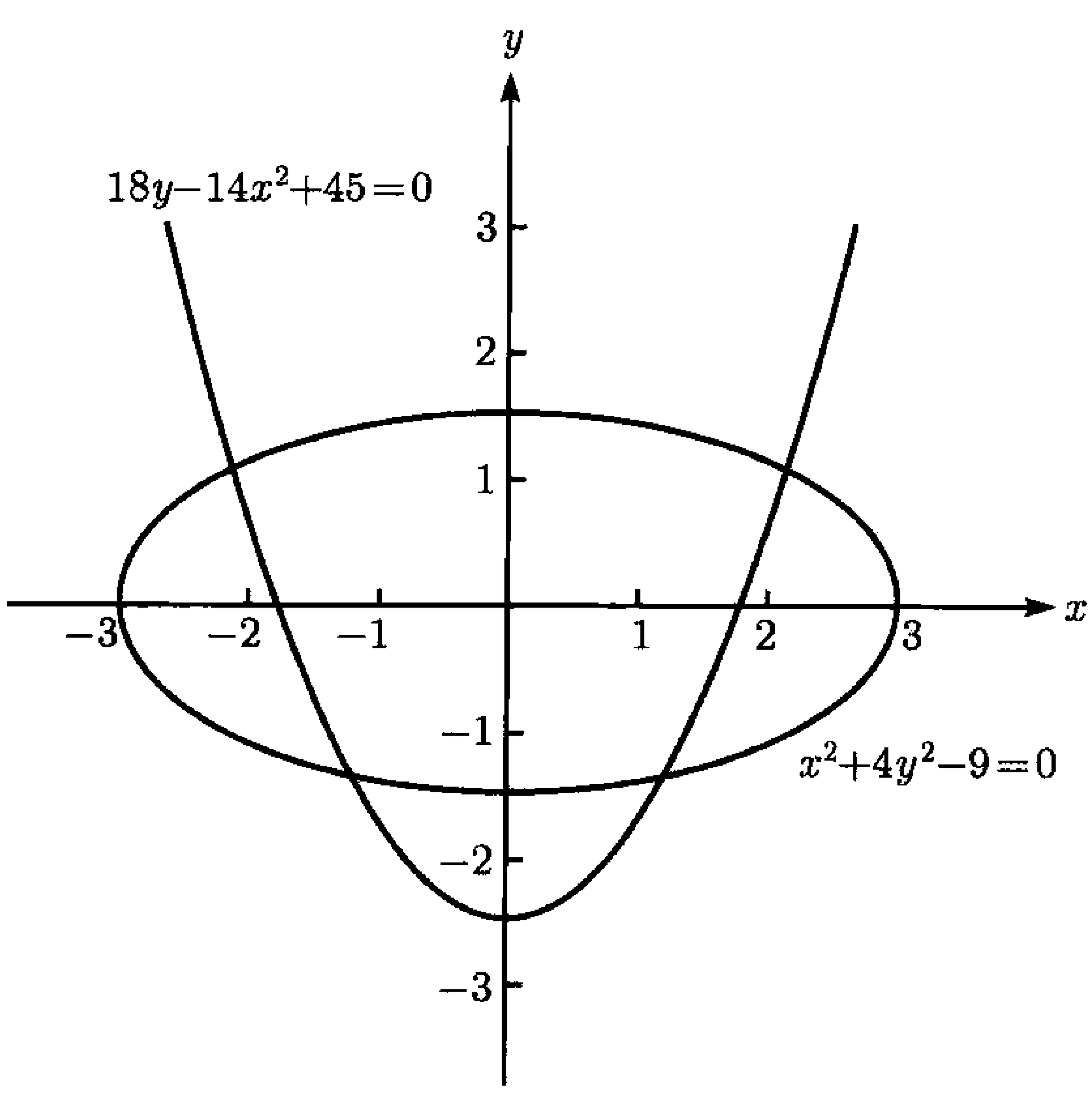


图 7-7　$f(x,y)=0$ 和 $g(x,y)=0$ 的图形

7.3.1　牛顿法

为了推导求解 (7.53) 的迭代法，我们推广 3.2 节的牛顿法. 首先复习多元微积

分中的切平面逼近.

设 (x_0, y_0) 是方程组 (7.53) 的解 $\alpha = (\xi, \eta)$ 的初始猜测. $z = f(x, y)$ 的图形是 xyz 空间内的一个曲面, 我们用在 $(x_0, y_0, f(x_0, y_0))$ 处与它相切的平面逼近它. 这个切平面的方程是 $z = p(x, y)$, 它可以表示成

$$p(x,y) \equiv f(x_0,y_0) + (x-x_0)f_x(x_0,y_0) + (y-y_0)f_y(x_0,y_0) \tag{7.54}$$

用记号

$$f_x(x,y) = \frac{\partial f(x,y)}{\partial x}, \quad f_y(x,y) = \frac{\partial f(x,y)}{\partial y}$$

分别表示 f 关于 x 和 y 的偏导数. 如果 $f(x_0, y_0)$ 充分接近于零, 则 $p(x, y)$ 的零曲线逼近 (x_0, y_0) 附近那些点 (x, y) 处的 $f(x, y)$ 的零曲线. 因为 $z = p(x, y)$ 的图形是一个平面, 所以它的零曲线就是一条直线.

例 7.3.1 考虑 (7.52) 中的函数 $f(x, y) \equiv x^2 + 4y^2 - 9$. 易得,

$$f_x(x,y) = 2x, \quad f_y(x,y) = 8y$$

在 $(x_0, y_0) = (1, -1)$ 处,

$$f(x_0,y_0) = -4, \quad f_x(x_0,y_0) = 2, \quad f_y(x_0,y_0) = -8$$

曲面 $z = f(x, y)$ 在 $(1, -1, -4)$ 处的切平面方程为

$$z = p(x,y) \equiv -4 + 2(x-1) - 8(y+1)$$

图 7-8 中给出 $f(x, y)$ 和 $p(x, y)$ 在 (x_0, y_0) 附近的 (x, y) 零曲线图形. [注意在此图中 $f(x, y) = 0$ 的图形只对应图 7-7 中一部分 $f(x, y) = 0$ 图形.] ■

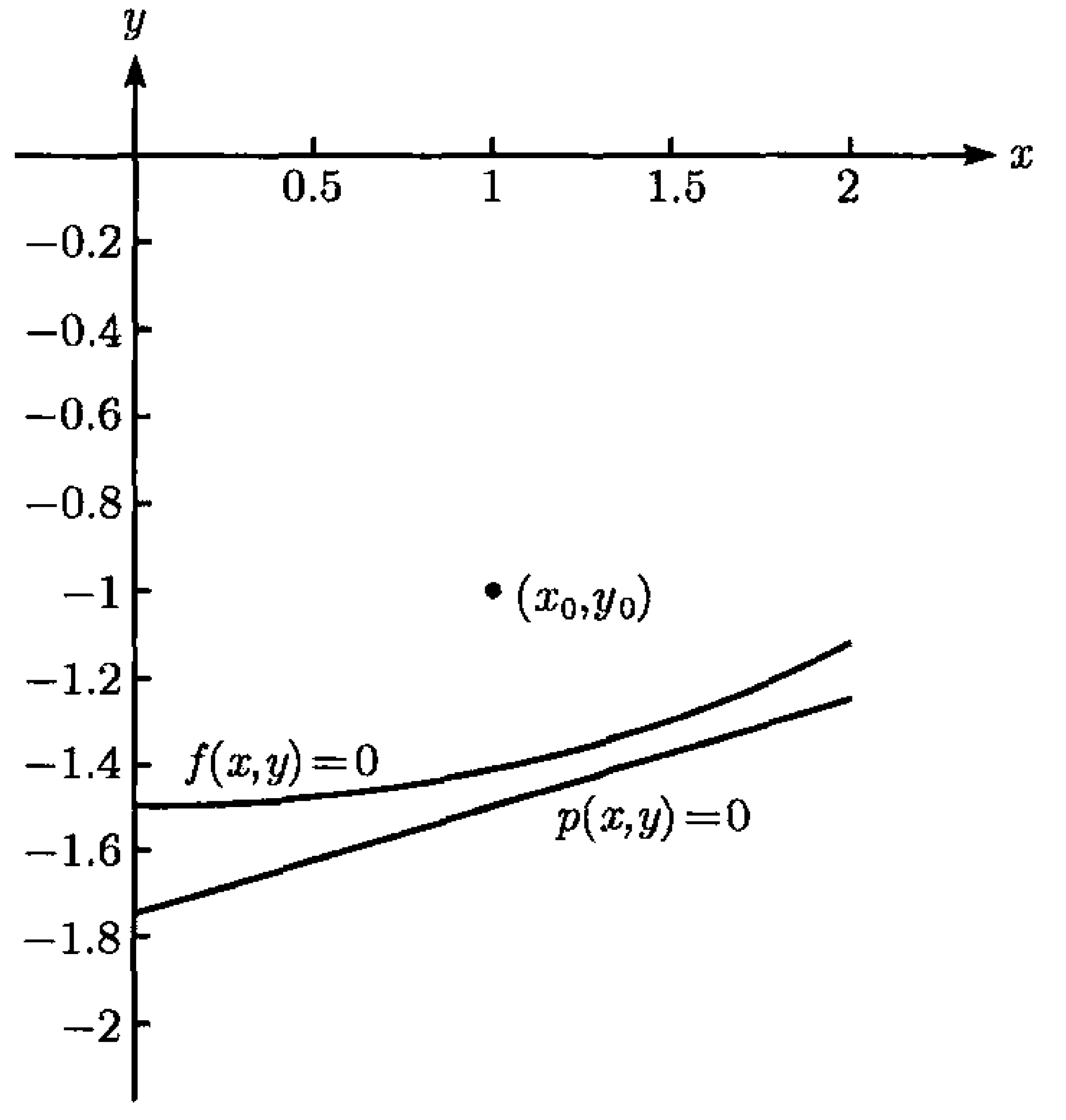


图 7-8 $f(x, y) = 0$ 和 $p(x, y) = 0$

上面切平面的构造也能应用于曲面 $z=g(x,y)$, 用 $z=q(x,y)$ 表示 $(x_0,y_0,g(x_0,y_0))$ 处的切平面方程:

$$q(x,y)\equiv g(x_0,y_0)+(x-x_0)g_x(x_0,y_0)+(y-y_0)g_y(x_0,y_0)$$

方程组 (7.53) 的解是 $z=f(x,y)$ 和 $z=g(x,y)$ 的零曲线的交集. 用切平面 $z=p(x,y)$ 和 $z=q(x,y)$ 的那些零曲线逼近这些零曲线, 并且用这些后来得到的零曲线的交集作为方程组 (7.53) 的近似解. 关于 $(x_0,y_0)=(1,-1)$ 附近方程组 (7.52) 的解, 在图 7-9 中给出说明.

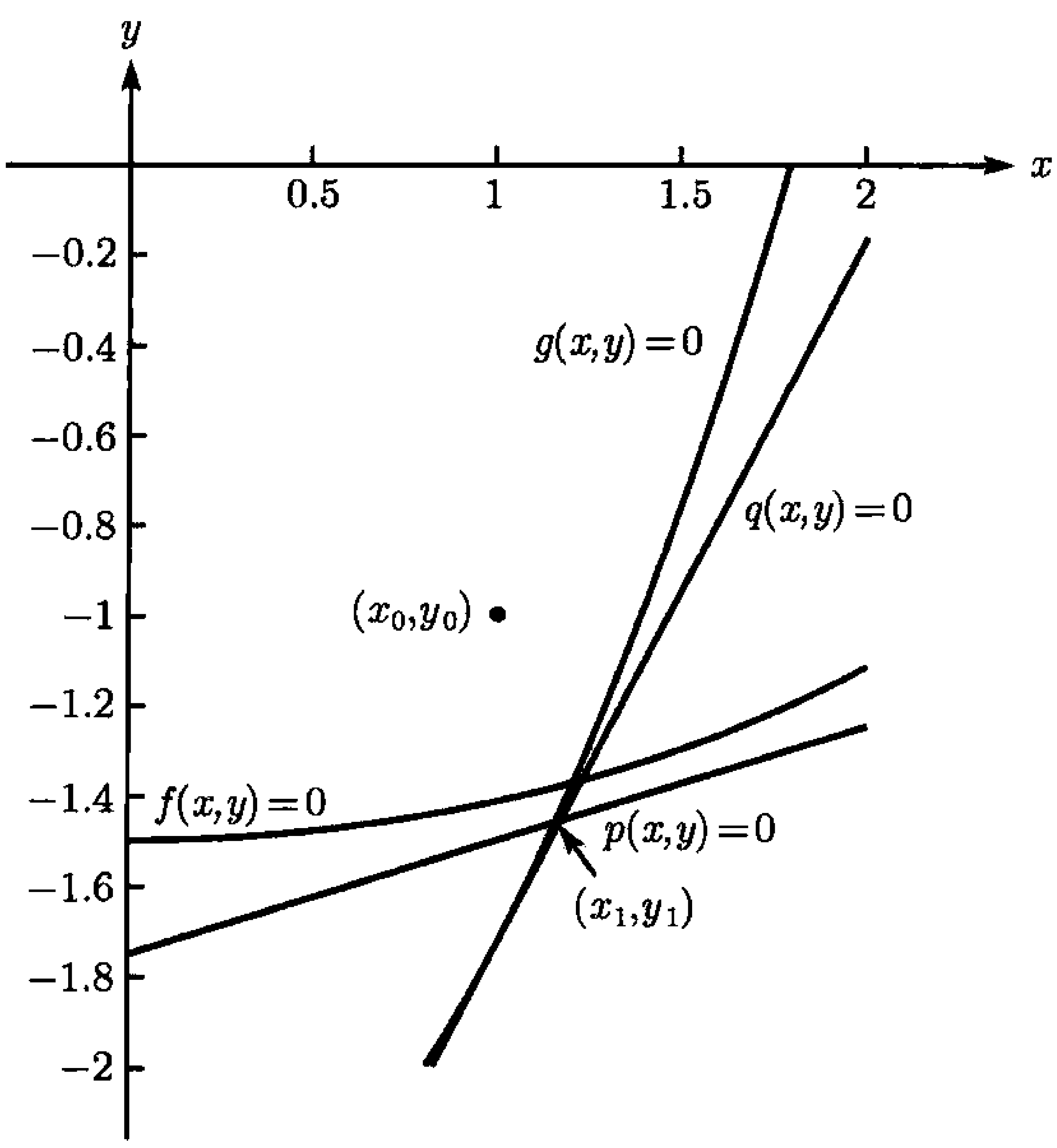


图 7-9　$f=g=0$ 和 $p=q=0$

为了求切平面的零曲线的交集, 必须求解线性方程组

$$\begin{cases} f(x_0,y_0)+(x-x_0)f_x(x_0,y_0)+(y-y_0)f_y(x_0,y_0)=0 \\ g(x_0,y_0)+(x-x_0)g_x(x_0,y_0)+(y-y_0)g_y(x_0,y_0)=0 \end{cases}$$

按矩阵形式, 它可以写成

$$\begin{bmatrix} f_x(x_0,y_0) & f_y(x_0,y_0) \\ g_x(x_0,y_0) & g_y(x_0,y_0) \end{bmatrix}\begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix}=-\begin{bmatrix} f(x_0,y_0) \\ g(x_0,y_0) \end{bmatrix}$$

我们用 (x_1,y_1) 表示这个方程组的解. 实际计算如下. 定义 δ_x 和 δ_y 为线性方程组

$$\begin{bmatrix} f_x(x_0,y_0) & f_y(x_0,y_0) \\ g_x(x_0,y_0) & g_y(x_0,y_0) \end{bmatrix}\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}=-\begin{bmatrix} f(x_0,y_0) \\ g(x_0,y_0) \end{bmatrix}$$

的解, 并且定义新的迭代

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}$$

通常, 点 (x_1, y_1) 比原来的点 (x_0, y_0) 更接近于解 $\boldsymbol{\alpha}$. 用 (x_1, y_1) 作为一个新的初始猜测, 重复这个过程, 可得到一个改进的估计 (x_2, y_2); 继续这个迭代过程直到得到的解具有足够的精度. 对于 $k = 0, 1, \cdots$, 一般的迭代公式为

$$\begin{bmatrix} f_x(x_k, y_k) & f_y(x_k, y_k) \\ g_x(x_k, y_k) & g_y(x_k, y_k) \end{bmatrix} \begin{bmatrix} \delta_{x,k} \\ \delta_{y,k} \end{bmatrix} = - \begin{bmatrix} f(x_k, y_k) \\ g(x_k, y_k) \end{bmatrix} \tag{7.55}$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} \delta_{x,k} \\ \delta_{y,k} \end{bmatrix} \tag{7.56}$$

这就是求解 (7.53) 的**牛顿法**.

例 7.3.2 如果我们再考虑方程组 (7.52), 则牛顿法 (7.55~7.56) 变为

$$\begin{bmatrix} 2x_k & 8y_k \\ -28x_k & 18 \end{bmatrix} \begin{bmatrix} \delta_{x,k} \\ \delta_{y,k} \end{bmatrix} = - \begin{bmatrix} x_k^2 + 4y_k^2 - 9 \\ 18y_k - 14x_k^2 + 45 \end{bmatrix} \tag{7.57}$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} \delta_{x,k} \\ \delta_{y,k} \end{bmatrix} \tag{7.58}$$

用 $(x_0, y_0) = (1, -1)$ 作为初始猜测来计算这个方程组的解, 得到解 (x_1, y_1). 重复这个过程得到迭代 (x_k, y_k), $k = 1, 2, \cdots$. 表 7-5 中给出前面几个迭代以及误差

$$\text{误差} = \|\boldsymbol{\alpha} - (x_k, y_k)\| \equiv \max\{|\xi - x_k|, |\eta - y_k|\}$$

最后的迭代精确到计算机算术运算的精度. ■

表 7-5 方程组 (7.52) 的牛顿迭代

k	x_k	y_k	误差
0	1.0	−1.0	3.74E−1
1	1.170 212 765 957 447	−1.457 446 808 510 638	8.34E−2
2	1.202 158 829 506 705	−1.376 760 321 923 060	2.68E−3
3	1.203 165 807 091 535	−1.374 083 486 949 713	2.95E−6
4	1.203 166 963 346 410	−1.374 080 534 243 534	3.59E−12
5	1.203 166 963 347 774	−1.374 080 534 239 942	2.22E−16

7.3.2 一般方程组的牛顿法

为了给出多于两个未知量的非线性方程组牛顿法的形式，我们首先改变求解两个未知量的方程组所使用的记号. 用

$$\begin{aligned} F_1(x_1, x_2) &= 0 \\ F_2(x_1, x_2) &= 0 \end{aligned} \tag{7.59}$$

取代 (7.53). 引进符号

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad F(\boldsymbol{x}) = \begin{bmatrix} F_1(x_1, x_2) \\ F_2(x_1, x_2) \end{bmatrix}$$

$$F'(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial F_1}{\partial x_1} & \dfrac{\partial F_1}{\partial x_2} \\ \dfrac{\partial F_2}{\partial x_1} & \dfrac{\partial F_2}{\partial x_2} \end{bmatrix}$$

$F'(\boldsymbol{x})$ 称为 $F(\boldsymbol{x})$ 的**弗雷歇导数**，它是普通的单变量函数导数到多变量情况的推广. 现在方程组 (7.59) 可以写成

$$F(\boldsymbol{x}) = \mathbf{0} \tag{7.60}$$

这个方程的解用 $\boldsymbol{\alpha}$ 表示.

对于 $k = 0, 1, \cdots$，求解 (7.53) 的牛顿方法 (7.55~7.56) 变为

$$\begin{aligned} F'(\boldsymbol{x}^{(k)})\boldsymbol{\delta}^{(k)} &= -F(\boldsymbol{x}^{(k)}) \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \boldsymbol{\delta}^{(k)} \end{aligned} \tag{7.61}$$

对于 $k = 0, 1, \cdots$，这可以改写成更简单并且数学上等价的形式

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - [F'(\boldsymbol{x}^{(k)})]^{-1} F(\boldsymbol{x}^{(k)}) \tag{7.62}$$

这个公式常用于非线性方程组牛顿法的理论讨论和分析；而 (7.61) 则用于牛顿法的实际计算，因为通常情况下，求解线性方程组比求系数矩阵的逆矩阵花费较少的代价. 注意到 (7.62) 与 3.2 节中单个方程牛顿法 (3.11) 的相似性.

现在考虑 n 个非线性方程的方程组

$$\begin{aligned} F_1(x_1, \cdots, x_n) &= 0 \\ &\vdots \\ F_n(x_1, \cdots, x_n) &= 0 \end{aligned} \tag{7.63}$$

定义

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad F(\boldsymbol{x}) = \begin{bmatrix} F_1(x_1, \cdots, x_n) \\ \vdots \\ F_n(x_1, \cdots, x_n) \end{bmatrix}$$

$$F'(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial F_1}{\partial x_1} & \cdots & \dfrac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial F_n}{\partial x_1} & \cdots & \dfrac{\partial F_n}{\partial x_n} \end{bmatrix}$$

非线性方程组 (7.63) 可以如同 (7.60) 那样写成 $F(\boldsymbol{x}) = \mathbf{0}$; 由 (7.61) 或 (7.62) 给出的牛顿法, 正好与两个未知量的两个方程的情况一样. 我们用超切平面逼近 (7.63) 中每个函数 $z = F_i(\boldsymbol{x})$ 来推导解 (7.63) 的牛顿法. 由于这个推导本质上就是前面两个方程情况所用论证的推广, 所以这里略去不讲. (7.62) 的记号统一了对任何未知量个数 $n \geqslant 1$ 的有限维方程组的牛顿法. 它也能推广到解 "无限维的问题", 如非线性微分方程和积分方程, 我们在本书中不考虑这种推广.

对于方程组牛顿法的推导是 1.1 节一般准则 (1.10) 和 3.3 节一般准则 (3.34) 的另一个说明. 对非线性方程组 $F(\boldsymbol{x}) = \mathbf{0}$ 线性化, 从而用一个较简单的相近的求根问题替代它.

例 7.3.3 考虑对未知函数 $z(s)$ 求解积分方程

$$z(s) + 4\int_0^1 \cos(st)[z(t)]^2 \mathrm{d}t = 4, \quad 0 \leqslant s \leqslant 1 \tag{7.64}$$

用一个非线性方程组逼近这个方程, 该方程组的解在 $[0,1]$ 中选定的点处得到 $z(s)$ 的近似值. 对于任何整数 $n > 0$, 设 $h = 1/n$ 以及 $t_j = (j - \frac{1}{2})h$, $j = 1, 2, \cdots, n$. 回顾中点数值求积法 [参见 5.1 节习题 8]:

$$\int_0^1 f(t)\mathrm{d}t \approx h\sum_{j=1}^n f(t_j)$$

将它应用于 (7.64) 中的积分, 然后在每个点 $s = t_i$ 处计算等式的值. 这就得到近似值

$$z(t_i) + 4h\sum_{j=1}^n \cos(t_i t_j)[z(t_j)]^2 \approx 4, \quad i = 1, 2, \cdots, n$$

求解关于 n 个未知量 $x_1, \cdots, x_n$ 的方程组

$$x_i + 4h\sum_{j=1}^n \cos(t_i t_j)[x_j]^2 = 4, \quad i = 1, 2, \cdots, n \tag{7.65}$$

预计有关系式

$$z(t_i) \approx x_i, \quad i = 1, 2, \cdots, n$$

方程组 (7.65) 用牛顿法求解, 使用的初始猜测是

$$x_i^{(0)} = 1, \quad i = 1, 2, \cdots, n \tag{7.66}$$

我们仍用 $\boldsymbol{\alpha}$ 表示非线性方程组的真解. 因为牛顿法收敛速度快, 所以我们使用误差估计

$$\boldsymbol{\alpha}-\boldsymbol{x}^{(k)} \approx \boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}=\boldsymbol{\delta}^{(k)} \tag{7.67}$$

$\boldsymbol{\delta}^{(k)}$ 是 (7.61) 中得到的修正. 表 7-6 给出用牛顿法求解 (7.65) 估计的迭代误差. 迭代 $\boldsymbol{x}^{(4)}$ 已精确到计算机算术运算的精度. 图 7-10 给出积分方程 (7.64) 的真解 $z(s)$, 以及对 $n=5$, 求解 (7.65) 得到的近似解. 更大的 n 值会给出更接近于 $z(t_i)$ 的 x_i, $i=1,2,\cdots,n$. ■

表 7-6　关于 $n=5$ 求解 (7.65) 的迭代误差

k	$\lVert\boldsymbol{\delta}^{(k)}\rVert$
0	3.34E−1
1	3.39E−2
2	4.29E−4
3	8.98E−8
4	3.53E−15

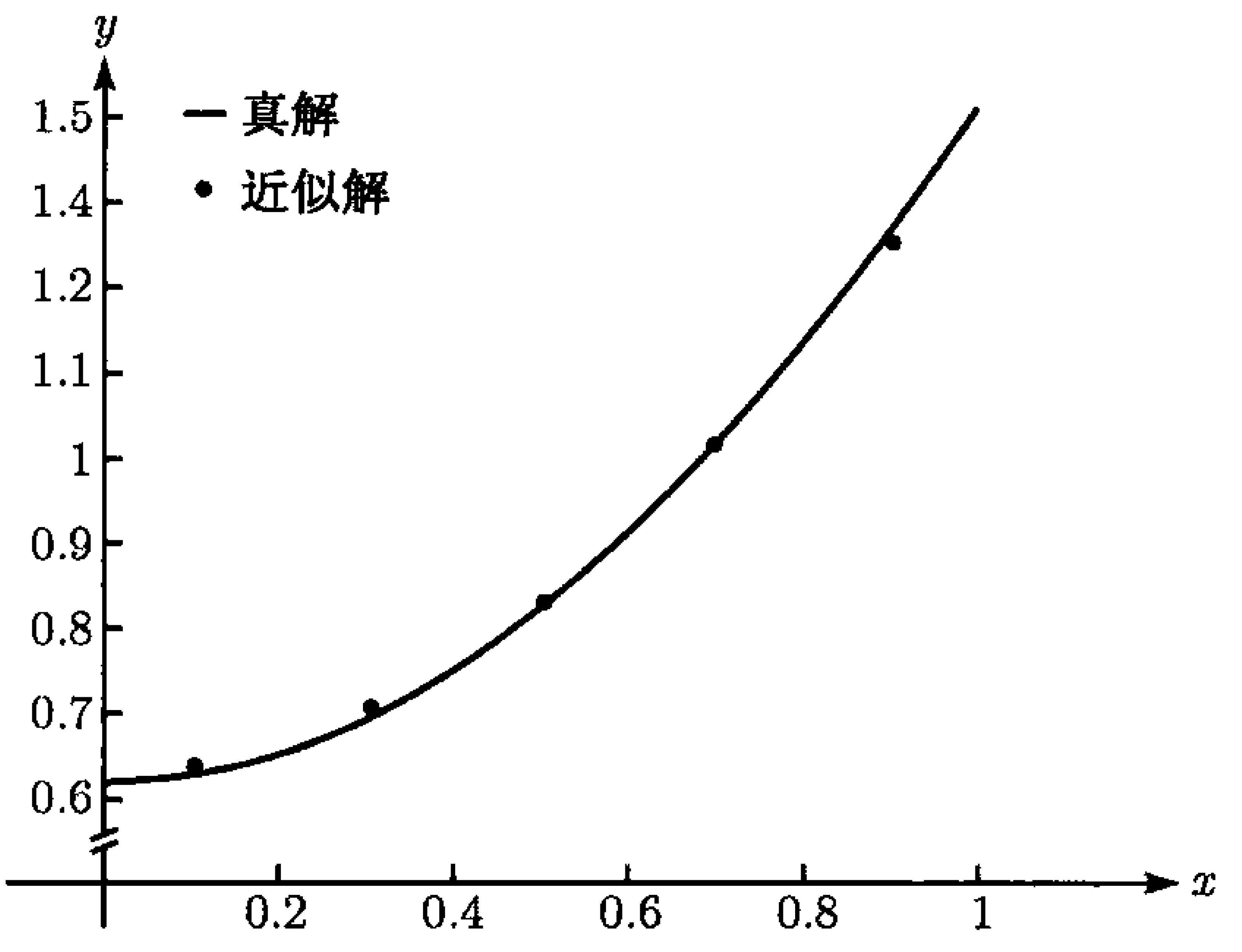


图 7-10　(7.64) 的真解和近似解

求解非线性方程组的牛顿法的收敛性分析相当复杂并且还在研究之中, 我们仅叙述主要结论而不加以证明.

定理 7.3.4　设 $\boldsymbol{\alpha}$ 是非线性方程组 $F(\boldsymbol{x})=\mathbf{0}$ 的一个解, 见 (7.63). 假设对于某个 $\varepsilon>0$ 以及所有 $i=1,2,\cdots,n$, 所有的点 $\boldsymbol{x}$ 满足 $\lVert\boldsymbol{\alpha}-\boldsymbol{x}\rVert<\varepsilon$, 函数 $F_i(x_1,\cdots,x_n)$ 对所有的变量 $x_1,\cdots,x_n$ 有连续的一阶偏导数和二阶偏导数. 进一步假设

$$\det F'(\boldsymbol{\alpha}) \neq 0 \tag{7.68}$$

如果选择 $\boldsymbol{x}^{(0)}$ 充分接近于 $\boldsymbol{\alpha}$, 则 (7.62) 定义的牛顿迭代 $\boldsymbol{x}^{(k)}$ 将收敛于 $\boldsymbol{\alpha}$. 另外, 对于适当选择的常数 $c>0$, 有

$$\left\|\boldsymbol{\alpha}-\boldsymbol{x}^{(k+1)}\right\| \leqslant c\|\boldsymbol{\alpha}-\boldsymbol{x}^{(k)}\|^2, \quad k=0,1,\cdots \tag{7.69}$$

这个定理推广了 3.2 节中用牛顿法求解单个非线性方程的讨论 [参见 (3.19) 及其下面的内容]. 条件 (7.68) 推广了条件 $f'(\alpha)\neq 0$, 它是前面 3.2 节中平方收敛分析要求的条件. 结论 (7.69) 说明牛顿法是平方收敛的. 用 (7.69), 可以证明

$$\lim_{k\to\infty}\frac{\|\boldsymbol{\alpha}-\boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\|}=1 \tag{7.70}$$

这是前面误差估计 (7.67) 的部分论证.

7.3.3 修正牛顿法

现在考虑用牛顿法 (7.61) 求解 n 个未知量的方程组 $F(\boldsymbol{x})=\boldsymbol{0}$ 所花费的代价. 每步迭代必须计算导数矩阵 $F'(\boldsymbol{x}^{(k)})$, 因为矩阵通常包含 n^2 个非零元素, 所以花费的代价是 n^2 次的函数计算. 另外, 必须求解线性方程组 (7.61), 每步迭代花费的代价大约是 $\frac{2}{3}n^3$ 次算术运算. 因为运算量大, 所以为了减少所花费的代价通常修正牛顿法 (7.61). 我们将给出一种这样的修正.

当 $\boldsymbol{x}^{(k)}$ 接近于 $\boldsymbol{\alpha}$ 时, 矩阵 $F'(\boldsymbol{x}^{(k)})$ 变化很小. 在计算机程序中, 我们尝试检查什么时候 $\boldsymbol{x}^{(k)}$ 充分接近于 $\boldsymbol{\alpha}$, 然后固定 (7.61) 中相应的导数矩阵. 为了使记号简单, 用 $\boldsymbol{A}$ 表示一个 "接近 $F'(\boldsymbol{\alpha})$" 的矩阵. 然后定义**修正牛顿法**为

$$\begin{aligned}\boldsymbol{A}\boldsymbol{\delta}^{(k)}&=-F(\boldsymbol{x}^{(k)})\\ \boldsymbol{x}^{(k+1)}&=\boldsymbol{x}^{(k)}+\boldsymbol{\delta}^{(k)}\end{aligned} \tag{7.71}$$

$k=0,1,\cdots$. 常常选择矩阵 $\boldsymbol{A}$ 为 $F'(\boldsymbol{x}^{(0)})$, 其中选择 $\boldsymbol{x}^{(0)}$ 充分接近于 $\boldsymbol{\alpha}$.

使用 (7.71) 花费多少代价? 其收敛的速度怎么样? 与牛顿法每步迭代计算 $F'(\boldsymbol{x})$ 对比, 一旦选择了 $\boldsymbol{A}$, 得到它不再进一步花费代价. 另外, 可以作 $\boldsymbol{A}$ 的 $\boldsymbol{LU}$ 分解, 一次花费算术运算的代价大约为 $\frac{2}{3}n^3$. 在随后的迭代中, 每步迭代求解 (7.71) 中线性方程组, 花费算术运算的代价大约仅为 $2n^2$. 对于大的 n, 这可以大大减少整个迭代花费的代价.

修正牛顿法 (7.71) 的收敛速度有所降低. 特别地, 如果, (1) 满足定理 7.3.4 的假设, (2) 选择 $\boldsymbol{A}$ 充分接近 $F'(\boldsymbol{\alpha})$, (3) 选择 $\boldsymbol{x}^{(0)}$ 充分接近 $\boldsymbol{\alpha}$, 则可以证明迭代 $\boldsymbol{x}^{(k)}$ 收敛于 $\boldsymbol{\alpha}$. 而且, 对某个 $c(\boldsymbol{A})<1$,

$$\left\|\boldsymbol{\alpha}-\boldsymbol{x}^{(k+1)}\right\| \leqslant c(\boldsymbol{A})\left\|\boldsymbol{\alpha}-\boldsymbol{x}^{(k)}\right\|, \text{ 对于 } k=0,1,\cdots$$

这表明 $\boldsymbol{x}^{(k)}$ **线性收敛于** $\boldsymbol{\alpha}$. 常量 $c(\boldsymbol{A})$ 是线性收敛率的上界, 并且可以证明当 $\boldsymbol{A}$ 靠近 $F'(\boldsymbol{\alpha})$ 时, 常量 $c(\boldsymbol{A})$ 收敛于零. 由于需要计算更多次的迭代, 因此迭代收敛

比牛顿法慢. 但是一般来说, 当 n 较大时, 这个方法增加的代价与用原来的牛顿法 (7.61) 涉及的额外花费的代价相比是比较小的.

例 7.3.5 再考虑积分方程 (7.64) 和作为逼近的非线性方程组 (7.65). 我们求解这个方程组 (如前, $n=5$), 并且选择 $\boldsymbol{A}=F'(\boldsymbol{x}^{(0)})$, 如 (7.66) 中那样取 $\boldsymbol{x}^{(0)}$. 迭代收敛结果在表 7-7 中给出, 注意给出的真实误差. 标记 "比率" 的列给出相继两次估计误差之比. 比率的值表明在这个情况下迭代收敛有些不规则, 并且 $c(\boldsymbol{A})\geqslant 0.247$. ■

表 7-7 对 $n=5$, 求解非线性方程组 (7.65) 的修正牛顿法

k	$\|\alpha-x^{(k)}\|$	比率
0	3.58E−1	
1	3.42E−2	0.096
2	7.03E−3	0.206
3	6.83E−4	0.097
4	1.54E−4	0.225
5	2.09E−5	0.136
6	2.30E−6	0.110
7	5.68E−7	0.247
8	5.53E−8	0.097
9	7.24E−9	0.131

牛顿法 (7.62) 和修正牛顿法 (7.71)

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\boldsymbol{A}^{-1}F(\boldsymbol{x}^{(k)})$$

都是不动点迭代

$$\boldsymbol{x}^{(k+1)}=g(\boldsymbol{x}^{(k)}),\quad k\geqslant 0$$

的例子, 其中 $g(\boldsymbol{x})$ 是一个向量函数

$$g(\boldsymbol{x})=\begin{bmatrix} g_1(x_1,\cdots,x_n) \\ \vdots \\ g_n(x_1,\cdots,x_n) \end{bmatrix}$$

这种不动点迭代法的一般理论涉及定理 3.4.2 的一个加强形式, 这里我们不加以考虑 [例如, 见 Atkinson(1989,§2.10)].

MATLAB 程序 我们给出一个 MATLAB 程序, 应用牛顿法求解两个未知量的两个非线性方程的方程组

$$F_1(x_1,x_2)=0$$

$$F_2(x_1,x_2)=0$$

```
function solution = newton_sys(x_init,err_tol,max_iterates)
%
% The calling sequence for newton_sys.m is
% solution = newton_sys(x_init,err_tol,max_iterates)
% This solves a pair of two nonlinear equations in two unknowns,
%              f(x) = 0
% with f(x) a column vector of length 2.  The definition of f(x)
% is to be given below in the function named fsys; and you
% also need to give the Jacobian matrix for f(x) in the
% function named deriv_fsys.
%
% x_init is a vector of length 2, and it is an initial guess
% at the solution.
%
% The parameters err_tol and max_iterates are upper limits on
% the desired error in the solution and the maximum number of
% iterates to be computed.

% Initialization.
x0 = zeros(2,1);
for i=1:2
   x0(i) = x_init(i);
end

error = inf;
it_count = 0;

% Begin the main loop.
while error > err_tol & it_count < max_iterates
   it_count = it_count + 1;
   rhs = fsys(x0);
   A = deriv_fsys(x0);
   delta = A\rhs;
   x1 = x0 - delta;
   error = norm(delta,inf);
   % The following statement is an internal print to show
   % the course of the iteration.  It and the pause
   % statement following it can be commented out.
   [it_count x1' error]
   pause
   x0 = x1;
end

% Return with the solution.
solution = x1;
if it_count == max_iterates
   disp(' ')
```

```
  disp('*** Your answers may possibly not ...
%                   satisfy your error tolerance.')
end

%%%%%%%%%%% Definition of functions %%%%%%%%%%%%%%%%%
function f_val = fsys(x)
%
% The equations being solved are
%   x(1)^2 + 4*x(2)^2 - 9 = 0
%   18*x(2) - 14*x(1)^2 + 45 = 0

f_val = [x(1)^2+4*x(2)^2-9, 18*x(2)-14*x(1)^2+45]';

function df_val = deriv_fsys(x)
%
% This defines the Jacobian matrix for the function
% given in fsys

df_val = [2*x(1), 8*x(2); -28*x(1), 18];
```

习题

1. 求方程组 (7.52) 的另外三个解. 注意, 可以使用对称性减少求解的个数, 但必须从计算中发现可以减少的解的个数.
2. 使用牛顿法 (7.55~7.56), 求下列方程组的全部解.
   (a) $x^2+y^2=4,\ x^2-y^2=1$
   (b) $x^2+4y^2=4,\ y=x^2-0.4x-1.96$
   (c) $x^2+y^2=1,\ 2y=2x^3+x+1$
   (d) $x+y-2xy=0,\ x^2+y^2-2x+2y+1=0$
3. 对于各种初始猜测 $(x_0,y_0)=(1.2,2.5),(-2,2.5),(-1.2,-2.5),(2,-2.5)$, 求方程组

$$x^2+xy^3=9,\quad 3x^2y-y^3=4$$

   的全部解. 观察各种迭代收敛的根及迭代计算次数. 评论你的结果.
4. 对于 $n=10,15,20$, 求解非线性方程组 (7.65), 达到你的计算机全精度. 比较这些迭代需要的次数. 如图 7-10 中那样, 对每个 n 画出方程组解 $\boldsymbol{x}$ 的图像.
5. 使用修正牛顿法 (7.71) 求解 (7.52) 的根, 结果按表 7-7 中那样列出. 使用 $\boldsymbol{A}=F'(\boldsymbol{x}^{(0)})$.
6. 对于习题 2 中给出的方程组重复习题 5 中的工作.
7. 把求函数 $D(x_1,x_2)$ 最大值或最小值问题化为两个未知量两个方程的方程组求根问题. 假设函数 $D(x_1,x_2)$ 关于 x_1 和 x_2 都可微.
8. 应用习题 7 求函数

$$D(x_1,x_2)=x_1^4+x_1x_2+(1+x_2)^2$$

   的极小值. 使用牛顿法求解所得到的方程组. 选择各种初始估计 $\boldsymbol{x}^{(0)}$ 进行实验.

第 8 章　常微分方程数值解

微分方程是建立物理、生物科学以及工程学的模型时所使用的最重要的数学工具之一. 在本章中, 我们考虑求解常微分方程 (即具有一个独立变量的微分方程) 的数值方法. 偏微分方程 (即具有多个独立变量的微分方程) 的数值方法在第 9 章中讨论.

本章绝大部分情况下考虑的微分方程具有下列形式

$$Y'(x) = f(x, Y(x)), \quad x \geqslant x_0 \tag{8.1}$$

其中 $Y(x)$ 是要寻找的未知函数. 给定两个变量的函数 $f(x,z)$ 就定义了相应的微分方程, 8.1 节中给出一些例子. 这个方程称为*一阶微分方程*, 因为它包含未知函数的一阶导数而不包含高阶导数. 高阶微分方程可化为一个等价的一阶方程组, 而一阶方程的数值方法能直接推广到这个新的方程组. 本章的后面会讨论高阶方程的这种转化和一阶微分方程组的数值解.

8.1 节讨论常微分方程初值问题的理论, 并引入微分方程稳定性的概念. 8.2 节和 8.3 节给出求解 (8.1) 的最简单的数值方法 ——*欧拉方法*. 它并不是一个有效的数值方法, 但研究它有助于引入和理解许多重要思想. 8.4 节讨论在实际计算中具有较好数值稳定性的一些数值方法. 8.5 节和 8.6 节包含一些更成熟的、收敛更快速的方法. 8.7 节中讨论高阶方程组和一阶方程组的数值解. 本章的最后一节考虑二阶常微分方程边值问题的有限差分法.

8.1　常微分方程理论简介

对简单的微分方程, 是有可能求出解的解析表达式的. 例如, 给定函数 g, 最简单的方程

$$Y'(x) = g(x)$$

的通解是

$$Y(x) = \int g(x)\mathrm{d}x + c$$

其中 c 是一个任意的积分常数. 这里, $\int g(x)\mathrm{d}x$ 表示 g 的某个固定的不定积分. 通过指定 $Y(x)$ 在某个已知点上的值

$$Y(x_0) = Y_0$$

可以得到常数 c 的值以及相应的特解.

例 8.1.1 方程

$$Y'(x) = \sin(x)$$

的通解是

$$Y(x) = -\cos(x) + c$$

若指定条件

$$Y\left(\frac{\pi}{3}\right) = 2$$

则容易求出 $c = 2.5$. 因而, 所要求的解是

$$Y(x) = 2.5 - \cos(x)$$ ■

对更一般的方程

$$Y'(x) = f(x, Y(x))$$

通解也依赖于一个常数. 为说明这一点, 我们考虑某些可以解析求解的例子. 首先, 也是主要考虑的, 是一阶线性方程

$$Y'(x) = a(x)Y(x) + b(x) \tag{8.2}$$

假定给定的函数 $a(x)$ 和 $b(x)$ 连续. 对这个方程,

$$f(x, z) = a(x)z + b(x)$$

并且我们可以用**积分因子法**来求方程的通解.

我们通过对很有用的一个特例

$$Y'(x) = \lambda Y(x) + b(x), \quad x \geqslant x_0 \tag{8.3}$$

来说明积分因子法, 这里 λ 是一个给定的常数. 在方程 (8.3) 两边乘以积分因子 $e^{-\lambda x}$ 后可以得到

$$\frac{d}{dx}(e^{-\lambda x}Y(x)) = e^{-\lambda x}b(x)$$

两边从 x_0 到 x 积分, 得到

$$e^{-\lambda x}Y(x) = c + \int_{x_0}^{x} e^{-\lambda t}b(t)dt$$

故 (8.3) 的通解是

$$Y(x) = e^{\lambda x}\left[c + \int_{x_0}^{x} e^{-\lambda t}b(t)dt\right] = ce^{\lambda x} + \int_{x_0}^{x} e^{\lambda(x-t)}b(t)dt \tag{8.4}$$

其中 c 是一个任意的常数, 容易看出

$$c = \mathrm{e}^{-\lambda x_0} Y(x_0) \tag{8.5}$$

从上面的讨论中可以看到, 一阶方程 (8.1) 的通解通常依赖于一个任意的积分常数. 为挑选一个特解, 需要指定一个附加条件. 通常, 选取这样的条件为

$$Y(x_0) = Y_0 \tag{8.6}$$

在常微分方程 (8.1) 的许多应用中, 独立变量 x 扮演时间的角色, 而 x_0 可认为是初始时间. 故习惯上称 (8.6) 为初值条件. 微分方程 (8.1) 和初值条件 (8.6) 一起构成一个初值问题

$$\begin{cases} Y'(x) = f(x, Y(x)), & x \geqslant x_0 \\ Y(x_0) = Y_0 \end{cases} \tag{8.7}$$

线性方程 (8.3) 初值问题的解由公式 (8.4) 和 (8.5) 给出. 注意到在函数 $b(x)$ 连续的任何开区间上解都存在, 这是线性方程的一个性质. 一般线性方程 (8.2) 的初值问题在函数 $a(x)$ 和 $b(x)$ 连续的任何开区间上解存在. 下面将举例说明, 当 (8.1) 是非线性方程时, 即使右端函数 $f(x, y)$ 有任意阶的导数, 相应的初值问题的解也可能仅在一个较小的区间上存在.

例 8.1.2 (1) 通过直接计算, 容易验证, 方程

$$Y'(x) = -[Y(x)]^2 + Y(x)$$

有所谓的平凡解 $Y(x) \equiv 0$ 以及通解

$$Y(x) = \frac{1}{1 + c\mathrm{e}^{-x}} \tag{8.8}$$

其中 c 为任意的常数. 另外, 这个方程是所谓的可分离方程, 它的解可以用习题 4 中所述的标准方法求出. 为求方程满足 $Y(0) = 4$ 的解, 在 (8.8) 中令 $x = 0$,

$$4 = \frac{1}{1 + c}$$

$$c = -0.75$$

故初值问题的解是

$$Y(x) = \frac{1}{1 - 0.75\mathrm{e}^{-x}}, \quad x \geqslant 0$$

对一般的初值 $Y(0) = Y_0 \neq 0$, 解公式 (8.8) 中的常数为 $c = Y_0^{-1} - 1$. 若 $Y_0 > 0$, 则 $c > -1$, 并且对 $0 \leqslant x < \infty$ 存在解 $Y(x)$. 然而, 对 $Y_0 < 0$, 仅在有限区间 $[0, \ln(1 - Y_0^{-1}))$ 上存在解, $x = \ln(1 - Y_0^{-1})$ 是公式 (8.8) 中分母的零点.

(2) 考虑方程

$$Y'(x) = -[Y(x)]^2$$

它有平凡解 $Y(x) \equiv 0$ 和通解

$$Y(x) = \frac{1}{x+c} \tag{8.9}$$

其中 c 为任意的常数. 这个结果可用直接计算来验证或用习题 4 所述的方法来得到. 为求满足初值条件 $Y(0) = Y_0$ 的方程的解, 我们按 Y_0 的值分成几种情况.

- 若 $Y_0 = 0$, 则对任意的 $x \geqslant 0$, 初值问题的解是 $Y(x) \equiv 0$
- 若 $Y_0 \neq 0$, 则初值问题的解是

$$Y(x) = \frac{1}{x + Y_0^{-1}}$$

- 若 $Y_0 > 0$, 则对任意的 $x \geqslant 0$, 解都存在. 若 $Y_0 < 0$, 仅在区间 $[0, -Y_0^{-1})$ 上解存在.
- 作为旁注, 注意到对 $0 < Y_0 < 1$, 解 (8.8) 关于 $x \geqslant 0$ 递增, 而若 $Y_0 > 0$, 解 (8.9) 关于 $x \geqslant 0$ 递减.

(3) 由 (8.4) 得,

$$Y'(x) = \lambda Y(x) + \mathrm{e}^{-x}, \quad Y(0) = 1, \quad \lambda \neq -1$$

的解为

$$\begin{aligned} Y(x) &= \mathrm{e}^{\lambda x} + \int_0^x \mathrm{e}^{\lambda(x-t)}\mathrm{e}^{-t}\mathrm{d}t \\ &= \mathrm{e}^{\lambda x}\left\{1 + \frac{1}{\lambda+1}[1 - \mathrm{e}^{-(\lambda+1)x}]\right\} \end{aligned}$$

■

注意, 对一般的右端函数 $f(x, z)$, 通常不可能解析地求解初值问题 (8.7). 一个这样的例子是方程

$$Y' = \mathrm{e}^{-xY^4}$$

在这种情况下, 数值方法是计算解的唯一可能的方法. 而且, 即使微分方程可以解析地求解, 如 (8.4) 那样的解公式通常包含一般函数的积分. 大多数积分必须数值计算. 例如, 容易验证, 问题

$$\begin{cases} Y' = 2xY + 1, & x > 0 \\ Y(0) = 1 \end{cases}$$

的解是

$$Y(x) = \mathrm{e}^{x^2}\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t + \mathrm{e}^{x^2}$$

对这样的情况, 一开始就使用数值方法求解微分方程通常更有效.

8.1.1 一般可解性理论

在考虑数值方法前, 对微分方程 (8.1) 的性质作一些讨论是有用的. 下列众所周知的结果涉及初值问题 (8.7) 的解的存在性和唯一性.

定理 8.1.3 设 $f(x,z)$ 和 $\partial f(x,z)/\partial z$ 在初始点 (x_0,Y_0) 的某个领域内的一切点 (x,z) 处是 x 和 z 的连续函数. 则存在定义在某个区间 $[x_0-\alpha,x_0+\alpha]$ 上唯一的函数 $Y(x)$, 满足

$$
\begin{aligned}
&Y'(x)=f(x,Y(x)), \quad x_0-\alpha\leqslant x\leqslant x_0+\alpha\\
&Y(x_0)=Y_0
\end{aligned}
$$

定理中的数 α 依赖于初值问题 (8.7). 对某些方程, 例如 (8.3) 中具有连续函数 $b(x)$ 的线性方程, 解对任意的 x 存在, 可取 α 为 ∞. 对许多非线性方程, 解只能在有界的区间中存在. 例 8.1.2 中就是这样的例子.

例 8.1.4 考虑初值问题

$$Y'(x)=2x[Y(x)]^2, \quad Y(0)=1$$

这里,

$$f(x,z)=2xz^2, \quad \frac{\partial f(x,z)}{\partial z}=4xz$$

对一切 (x,z), 这两个函数连续. 因而, 由定理 8.1.3 知, 在 $x_0=0$ 的某个领域内, 这个初值问题存在唯一解. 这个解是

$$Y(x)=\frac{1}{1-x^2}, \quad -1<x<1$$

此例说明, $f(x,z)$ 和 $\partial f(x,z)/\partial z$ 的连续性并不保证存在对于所有的 x 连续的解. ■

8.1.2 初值问题的稳定性

数值求解初值问题 (8.7) 时, 一般会在给定的有限区间 $x_0\leqslant x\leqslant b$ 上寻找解 $Y(x)$. 在此情况下, 可以得到下列关于稳定性的结果. 对初值问题的初始值作一个小的改变, 把 Y_0 变成 $Y_0+\varepsilon$. 相应的解记为 $Y_\varepsilon(x)$, 它满足

$$Y_\varepsilon'(x)=f(x,Y_\varepsilon(x)), \quad x_0\leqslant x\leqslant b, \quad Y_\varepsilon(x_0)=Y_0+\varepsilon \tag{8.10}$$

在类似于定理 8.1.3 的那些假设之下, 可以证明, 对一切小的 ε 值,

$$\max_{x_0\leqslant x\leqslant b}|Y_\varepsilon(x)-Y(x)|\leqslant c\varepsilon, \quad c>0 \tag{8.11}$$

因而, 初值 Y_0 小的改变将导致初值问题的解 $Y(x)$ 小的改变. 对许多应用来说这是一个令人满意的性质.

例 8.1.5 问题

$$Y'(x) = -Y(x) + 1, \quad 0 \leqslant x \leqslant b, \quad Y(0) = 1 \tag{8.12}$$

有解 $Y(x) \equiv 1$. 扰动问题

$$Y_\varepsilon'(x) = -Y_\varepsilon(x) + 1, \quad 0 \leqslant x \leqslant b, \quad Y_\varepsilon(0) = 1 + \varepsilon$$

有解 $Y_\varepsilon(x) = 1 + \varepsilon \mathrm{e}^{-x}$, $x \geqslant 0$. 因而

$$Y(x) - Y_\varepsilon(x) = -\varepsilon \mathrm{e}^{-x}$$
$$|Y(x) - Y_\varepsilon(x)| \leqslant |\varepsilon|, \quad x \geqslant 0$$

称问题 (8.12) 是稳定的. ■

几乎所有的初值问题都具有 (8.11) 这样的稳定性, 但这不是初值 Y_0 小扰动影响的所有方面. 若 (8.11) 中最大的误差比 ε 大得多 [即估计式 (8.11) 中最小可能的常数 c 很大], 则通常初值问题 (8.7) 认为是*病态的*. 试图数值求解这样的问题通常将导致计算解大的误差.

例 8.1.6 问题

$$Y'(x) = \lambda[Y(x) - 1], \quad 0 \leqslant x \leqslant b, \quad Y(0) = 1 \tag{8.13}$$

有解

$$Y(x) = 1, \quad 一切 \ x$$

扰动问题

$$Y_\varepsilon'(x) = \lambda\left[Y_\varepsilon(x) - 1\right], \quad 0 \leqslant x \leqslant b, \quad Y_\varepsilon(0) = 1 + \varepsilon$$

有解

$$Y_\varepsilon(x) = 1 + \varepsilon \mathrm{e}^{\lambda x}, \quad x \geqslant 0$$

误差为

$$Y(x) - Y_\varepsilon(x) = -\varepsilon \mathrm{e}^{\lambda x}$$

$$\max_{0 \leqslant x \leqslant b} |Y(x) - Y_\varepsilon(x)| = \begin{cases} |\varepsilon|, & \lambda \leqslant 0 \\ |\varepsilon| \mathrm{e}^{\lambda b}, & \lambda \geqslant 0 \end{cases} \tag{8.14}$$

若 $\lambda < 0$, 误差 $|Y(x) - Y_\varepsilon(x)|$ 随着 x 增大而减小. 当 $\lambda \leqslant 0$ 时 (8.13) 是良态的. 相反, 若 $\lambda > 0$, 误差 $|Y(x) - Y_\varepsilon(x)|$ 随着 x 增大而增大. 即使对中等大小的 λb, 比如 $\lambda b \geqslant 10$, 在 $x = b$ 处解 $Y(x)$ 的改变也已经相当显著. 随着 λ 的增大, 问题 (8.13) 变得越来越病态. ■

对更一般的初值问题 (8.7) 以及扰动问题 (8.10), 可以证明, 对充分接近 x_0 的 x,

$$Y(x) - Y_\varepsilon(x) \approx -\varepsilon \exp\left(\int_{x_0}^{x} g(t)\mathrm{d}t\right) \tag{8.15}$$

其中

$$g(t) = \left.\frac{\partial f(t,z)}{\partial z}\right|_{z=Y(t)}$$

注意到这个公式正确地预言了 (8.14), 因为对那个问题

$$f(x,z) = \lambda[z-1], \quad \frac{\partial f(x,z)}{\partial z} = \lambda, \quad \int_0^x g(t)\mathrm{d}t = \lambda x$$

接着由 (8.15) 得到

$$Y(x) - Y_\varepsilon(x) \approx -\varepsilon \mathrm{e}^{\lambda x}$$

这与前面的公式 (8.14) 一致.

例 8.1.7 问题

$$Y'(x) = -[Y(x)]^2, \quad Y(0) = 1 \tag{8.16}$$

有解

$$Y(x) = \frac{1}{x+1}$$

对扰动问题

$$Y_\varepsilon'(x) = -[Y_\varepsilon(x)]^2, \quad Y_\varepsilon(0) = 1+\varepsilon \tag{8.17}$$

我们利用 (8.15) 来估计 $Y(x) - Y_\varepsilon(x)$. 首先

$$f(x,z) = -z^2, \quad \frac{\partial f(x,z)}{\partial z} = -2z$$

$$g(t) = -2Y(t) = -\frac{2}{t+1}$$

$$\int_0^x g(t)\mathrm{d}t = -2\int_0^x \frac{\mathrm{d}t}{t+1} = -2\ln(1+x) = \ln(1+x)^{-2}$$

$$\exp\left[\int_0^x g(t)\mathrm{d}t\right] = \mathrm{e}^{\ln(1+x)^{-2}} = \frac{1}{(x+1)^2}$$

代入 (8.15) 中, 对充分小的 $x \geqslant 0$, 有

$$Y(x) - Y_\varepsilon(x) \approx \frac{-\varepsilon}{(1+x)^2} \tag{8.18}$$

这表明 (8.16) 是一个良态问题. ■

一般来说, 若

$$\frac{\partial f(x,Y(x))}{\partial z} \leqslant 0, \quad x_0 \leqslant x \leqslant b \tag{8.19}$$

则称初值问题良态. 虽然这个检验依赖于区间 $[x_0,b]$ 上的解 $Y(x)$, 但是通常我们不用知道 $Y(x)$ 就可以证明 (8.19).

8.1.3　方向场

方向场是理解微分方程解的特性的一个有用的工具. 对方程 $Y'=f(x,Y)$, 在解曲线的任何点 (x,y) 处的斜率是 $f(x,y)$. 在方向场图中, 斜率可以用图形表示出来. 在 MATLAB 程序中, 可以利用meshgrid和quiver指令生成方向场.

例 8.1.8　考虑方程 $Y'=Y$. 解曲线在曲线上的点 (x,y) 处的斜率是 y, 它与 x 无关. 用下列 MATLAB 代码生成方向场图. 首先画方向场

```
[x,y] = meshgrid(-2:0.5:2,-2:0.5:2);
dx = ones(9);          %Generates a matrix of 1's.
dy = y; quiver(x,y,dx,dy);
```

然后画两条解曲线

```
hold on
x = -2:0.01:1;
y1 = exp(x);
y2 = -exp(x);
plot(x,y1,x,y2)
text(1.1,2.8,'\itY=e^x','FontSize',14)
text(1.1,2.8,'\itY=-e^x','FontSize',14)
hold off
```

结果如图 8-1 所示. ■

例 8.1.9　继续讨论例 8.1.4, 用下列 MATLAB 的 M– 文件, 生成图 8-2 中的方向场图和特解 $Y=1/(1-x^2)$.

```
[x,y] = meshgrid(-1:0.2:1,1:0.5:4);
dx = ones(7,11);
dy = 2*x.*y.^2;
quiver(x,y,dx,dy);
hold on
xx = -0.87:0.01:0.87;
yy = 1./(1-xx.^2);
plot(xx,yy)
hold off
```

注意, 对大的 y 值, 方向场图中的箭头指向几乎垂直, 这说明方程只可能在 x 轴的一个有界区间内有解. ■

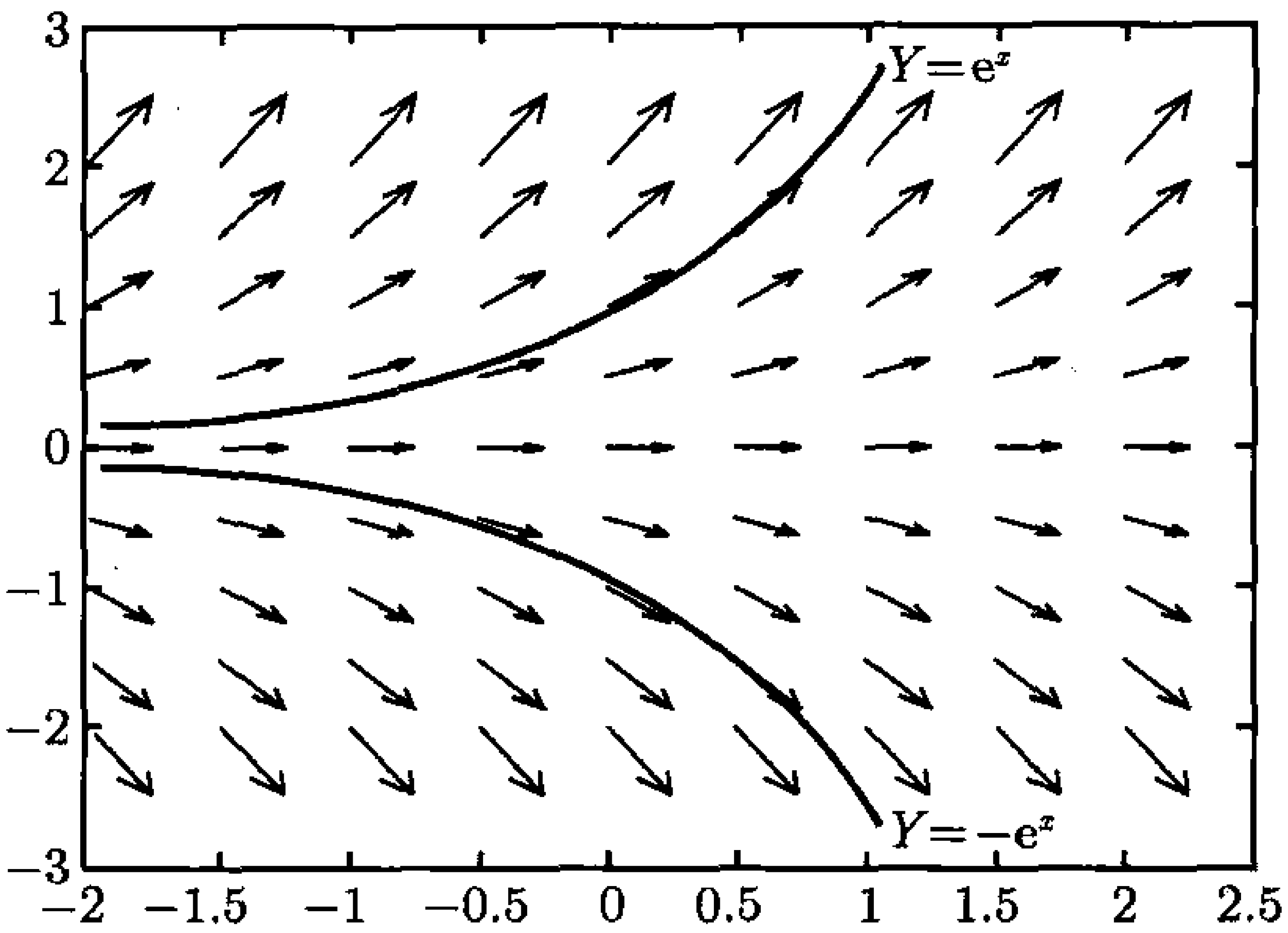


图 8-1 方程 $Y = Y'$ 的方向场 $Y = \pm e^x$

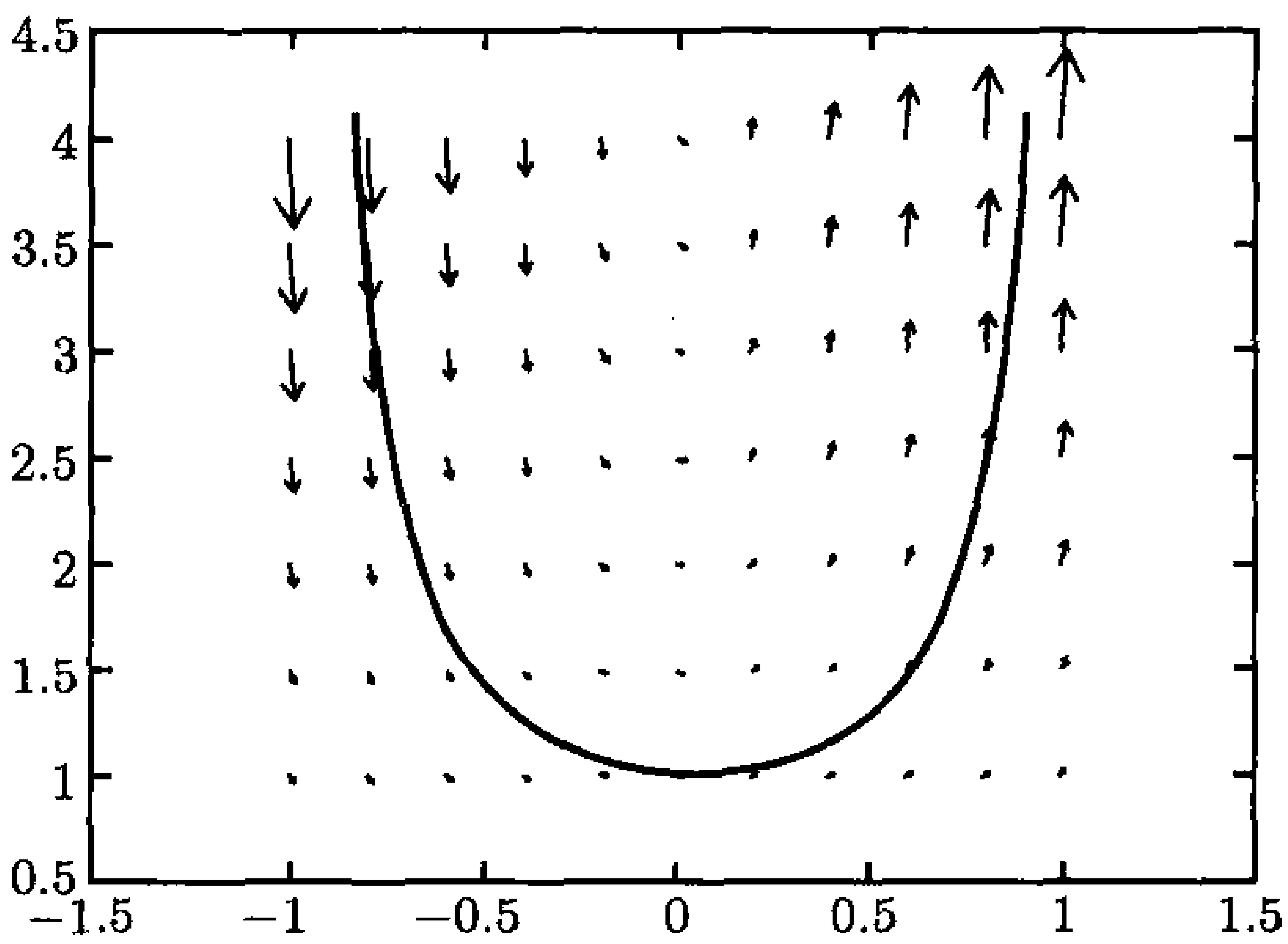


图 8-2 方程 $Y' = 2xY^2$ 的方向场和解 $Y = 1/(1 - x^2)$

习题

1. 对下列每种情况, 证明给出的函数 $Y(x)$ 满足相关的微分方程, 然后用初始条件确定常数 c 的值. 最后, 参考 (8.7) 中的一般格式, 对每个微分方程给出相应的 $f(x, z)$.

(a) $Y'(x) = -Y(x) + \sin(x) + \cos(x), \quad Y(0) = 1;\ Y(x) = \sin(x) + ce^{-x}$

(b) $Y'(x) = (Y(x) - Y(x)^2)/x, \quad Y(1) = 2;\ Y(x) = x/(x + c), \quad x > 0$

(c) $Y'(x) = \cos^2(Y(x)), \quad Y(0) = \pi/4;\ Y(x) = \tan^{-1}(x + c)$

(d) $Y'(x) = Y(x)[Y(x) - 1], \quad Y(0) = \dfrac{1}{2};\quad Y(x) = 1/(1 + ce^{-x})$

2. 利用 MATLAB 画出习题 1 中的微分方程的方向场.

3. 利用 (8.4) 解下列问题:

(a) $Y'(x) = \lambda Y(x) + 1, \quad Y(0) = 1$

(b) $Y'(x) = \lambda Y(x) + x, \quad Y(0) = 3$

4. 对给定的函数 $f_1(x)$ 和 $f_2(x)$, 考虑微分方程

$$Y'(x) = f_1(x)f_2(Y(x))$$

这个方程称为*可分离的微分方程*. 这种方程可以通过直接积分来求解. 把方程改写为

$$\frac{Y'(x)}{f_2(Y(x))} = f_1(x)$$

两边积分:

$$\int \frac{Y'(x)\mathrm{d}x}{f_2(Y(x))} = \int f_1(x)\mathrm{d}x$$

对左边的积分进行变量变换, 令 $z = Y(x)$. 则等式变成

$$\int \frac{\mathrm{d}z}{f_2(z)} = \int f_1(x)\mathrm{d}x$$

积分以后, 用 $Y(x)$ 代替 z; 如果可能的话, 再求解 $Y(x)$. 如果这些积分可以计算, 则可以得到微分方程的解. 用上述方法求下列问题的通解以及满足给定初始条件的特解.

(a) $Y'(x) = x/Y(x), \quad Y(0) = 2$

(b) $Y'(x) = x\mathrm{e}^{-Y(x)}, \quad Y(1) = 0$

(c) $Y'(x) = Y(x)[a - Y(x)], \quad Y(0) = a/2, \quad a > 0$

5. 讨论习题 1 中初值问题的性态. 利用检验式 (8.19).

6. 利用 (8.19) 讨论问题

$$Y'(x) = Y(x)^2 - 5\sin(x) - 25\cos^2(x), \quad Y(0) = 6$$

的性态. 你不必知道真解.

7. 对常数 a, b, d, 其中 $a, b > 0$, 求

$$Y'(x) + aY(x) = d\mathrm{e}^{-bx}$$

的解 $Y(x)$. 计算

$$\lim_{x\to\infty} Y(x)$$

提示: 分别考虑 $a \neq b$ 和 $a = b$ 的情况.

8.2 欧 拉 方 法

求解初值问题最简单的方法称为*欧拉方法*. 本节中定义这个方法, 然后在下节中分析它. 欧拉方法并不是最有效的数值方法, 但用它引入微分方程数值解中的许多想法最为简单.

首先, 建立本章将使用的一些记号. 如前, $Y(x)$ 表示初值问题对初值 Y_0 的真解:

$$\begin{aligned} &Y'(x) = f(x, Y(x)), \quad x_0 \leqslant x \leqslant b \\ &Y(x_0) = Y_0 \end{aligned} \tag{8.20}$$

求解 (8.20) 的数值方法将在一组离散的节点

$$x_0 < x_1 < x_2 < \cdots < x_N \leqslant b \tag{8.21}$$

上得到近似解 $y(x)$.

为简单起见, 取等距节点:

$$x_n = x_0 + nh, \quad n = 0, 1, \cdots, N$$

我们用 $y(x)$ 表示近似解. 节点上的近似解用下列记号表示:

$$y(x_n) = y_h(x_n) = y_n, \quad n = 0, 1, \cdots, N$$

为了得到 $[x_0, b]$ 中不同于 (8.21) 的那些点上的近似解 $y(x)$, 可用插值公式. 尽管容易应用第 4 章的技巧, 但是我们在此不考虑这个问题.

为导出欧拉方法, 回顾第 5 章的导数逼近 (5.76)

$$Y'(x) \approx \frac{1}{h}[Y(x+h) - Y(x)]$$

把此式用于 $x = x_n$ 处的初值问题 (8.20)

$$Y'(x_n) = f(x_n, Y(x_n))$$

得到

$$\begin{aligned} \frac{1}{h}[Y(x_{n+1}) - Y(x_n)] &\approx f(x_n, Y(x_n)) \\ Y(x_{n+1}) &\approx Y(x_n) + hf(x_n, Y(x_n)) \end{aligned} \tag{8.22}$$

让上式两边相等来定义欧拉方法:

$$y_{n+1} = y_n + hf(x_n, y_n), \quad 0 \leqslant n \leqslant N-1 \tag{8.23}$$

作为初值, 取 $y_0 = Y_0$ 或某个接近于 Y_0 的近似值. 有时 Y_0 的值是由实验得到的, 因而可能仅仅近似地知道这个值. 公式 (8.23) 给出逐次计算 $y_1, y_2, \cdots, y_N$ 的法则. 这是求解常微分方程最典型的数值方法.

图 8-3 给出欧拉方法的几何解释. 曲线 $z = Y(x)$ 的图形在 x_n 处的切线斜率为

$$Y'(x_n) = f(x_n, Y(x_n))$$

若使用这条切线逼近点 $(x_n, Y(x_n))$ 附近的曲线, 则 (8.22) 右边给出切线函数在 $x = x_{n+1}$ 处的值.

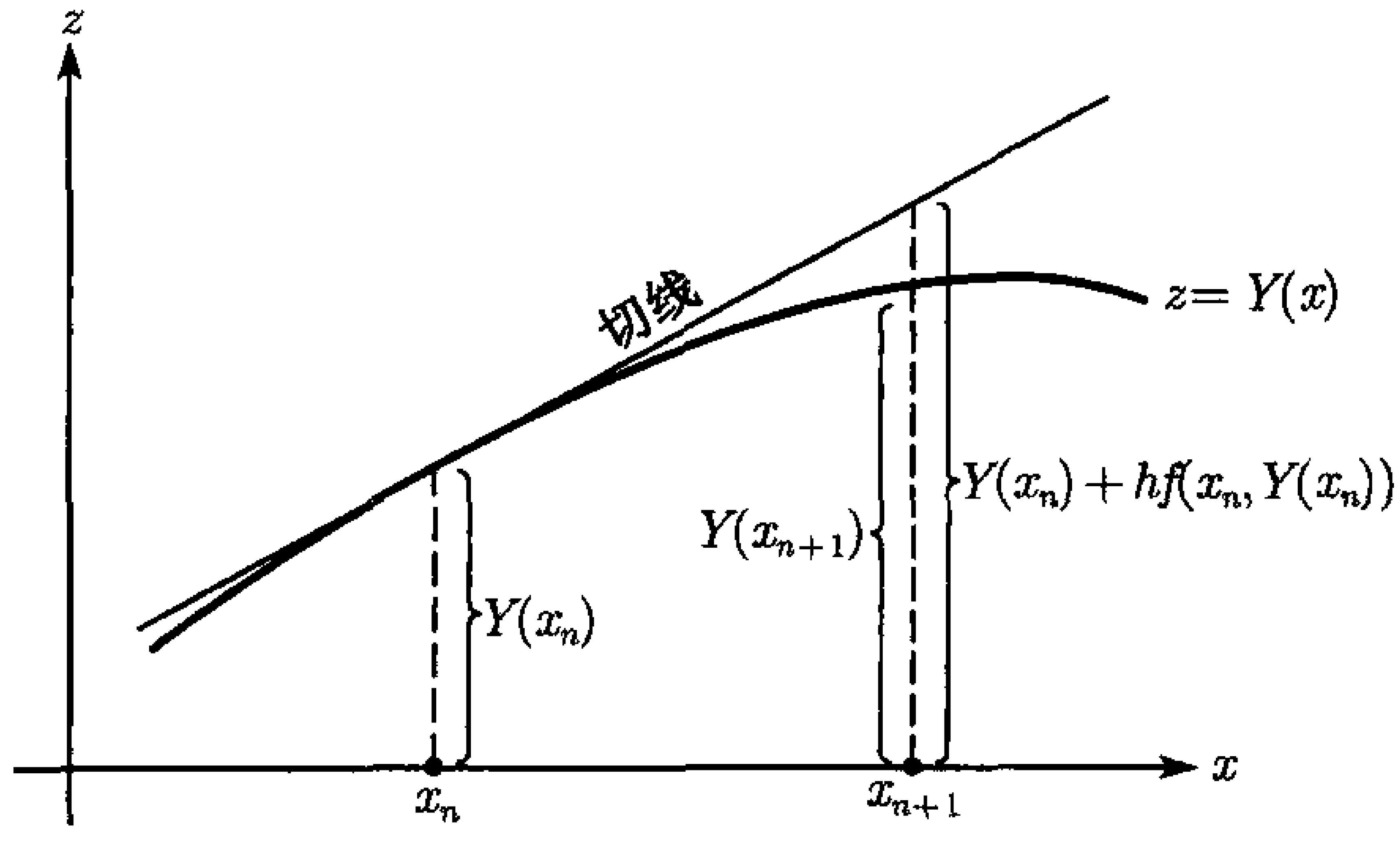


图 8-3 欧拉方法的几何解释

例 8.2.1 (a) 问题

$$Y'(x) = -Y(x), \quad Y(0) = 1 \tag{8.24}$$

的真解是 $Y(x) = \mathrm{e}^{-x}$. 下式给出欧拉方法

$$y_{n+1} = y_n - hy_n, \quad n \geqslant 0 \tag{8.25}$$

其中 $y_0 = 1$, $x_n = nh$. 表 8-1 对三个 h 值在一些 x 值处给出解 $y(x)$. 为说明计算过程, 我们来求 $h = 0.1$ 时的 y_1 和 y_2. 由 (8.25),

表 8-1 (8.24) 的欧拉方法

h	x	$y_h(x)$	误差	相对误差
0.2	1.0	3.276 8E−1	4.02E−2	0.109
	2.0	1.073 8E−1	2.80E−2	0.207
	3.0	3.518 4E−2	1.46E−2	0.293
	4.0	1.152 9E−2	6.79E−2	0.371
	5.0	3.777 9E−3	2.96E−3	0.439
0.1	1.0	3.486 7E−1	1.92E−2	0.052 2
	2.0	1.215 8E−1	1.38E−2	0.102
	3.0	4.239 1E−2	7.40E−3	0.149
	4.0	1.478 1E−2	3.53E−3	0.193
	5.0	5.153 8E−3	1.58E−3	0.234
0.05	1.0	3.584 9E−1	9.39E−3	0.025 5
	2.0	1.285 1E−1	6.82E−3	0.050 4
	3.0	4.607 0E−2	3.72E−3	0.074 7
	4.0	1.651 5E−2	1.80E−3	0.098 3
	5.0	5.920 5E−3	8.17E−4	0.121

$$y_1 = y_0 - hy_0 = 1 - (0.1)(1) = 0.9, \quad x_1 = 0.1$$
$$y_2 = y_1 - hy_1 = 0.9 - (0.1)(0.9) = 0.81, \quad x_2 = 0.2$$

这些数值解的误差为

$$Y(x_1) - y_1 = \mathrm{e}^{-0.1} - y_1 \doteq 0.004\,837$$
$$Y(x_2) - y_2 = \mathrm{e}^{-0.2} - y_2 \doteq 0.008\,731$$

(b) 问题

$$Y'(x) = \frac{Y(x) + x^2 - 2}{x+1}, \quad Y(0) = 2 \tag{8.26}$$

的真解是

$$Y(x) = x^2 + 2x + 2 - 2(x+1)\ln(x+1)$$

这个微分方程的欧拉方法是

$$y_{n+1} = y_n + \frac{h(y_n + x_n^2 - 2)}{x_n + 1}, \quad n \geqslant 0$$

其中 $y_0 = 2$, $x_n = nh$. 表 8-2 对 h 的三个值和一些 x 值给出解 $y(x)$. 图 8-4 中给出 $h = 0.2$ 的解 $y_h(x)$ 的图形. 图中用直线段连接节点值 $y_h(x_n)$. 注意水平尺度和垂直尺度是不同的. ■

表 8-2 (8.26) 的欧拉方法

h	x	$y_h(x)$	误差	相对误差
0.2	1.0	2.159 2	6.82E−2	0.030 6
	2.0	3.169 7	2.39E−1	0.070 1
	3.0	5.433 2	4.76E−1	0.080 5
	4.0	9.141 1	7.65E−1	0.129
	5.0	14.406	1.09	0.070 3
	6.0	21.303	1.45	0.063 7
0.1	1.0	2.191 2	3.63E−2	0.016 3
	2.0	3.284 1	1.24E−1	0.036 4
	3.0	5.663 6	2.46E−1	0.041 6
	4.0	9.512 5	3.93E−1	0.066 5
	5.0	14.939	5.60E−1	0.036 1
	6.0	22.013	7.44E−1	0.032 7
0.05	1.0	2.208 7	1.87E−2	0.008 40
	2.0	3.344 9	6.34E−2	0.018 6
	3.0	5.784 5	1.25E−1	0.021 2
	4.0	9.706 1	1.99E−1	0.033 7
	5.0	15.214	2.84E−1	0.018 3
	6.0	22.381	3.76E−1	0.016 5

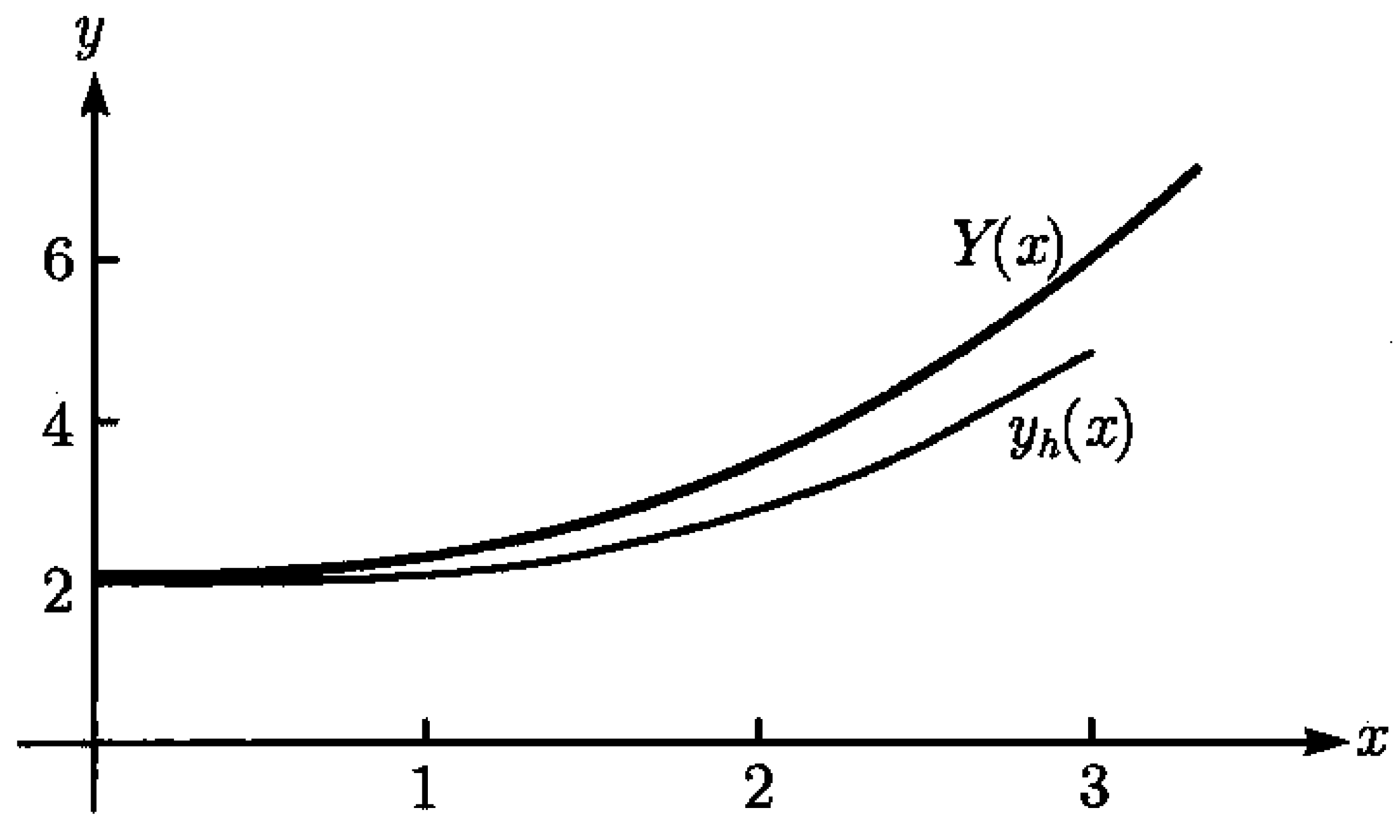


图 8-4 问题 (8.26) 的欧拉方法, $h = 0.2$

我们来考察在两例中 h 递减时误差变化的性态. 对每个固定的 x 值, 当 h 折半时误差大约减小到原来的 1/2. 例如, 在例 8.2.1(a) 中取 $x = 5.0$. 当 $h = 0.2, 0.1, 0.05$ 时误差分别是

$$2.96\mathrm{E}-3, \quad 1.58\mathrm{E}-3, \quad 8.17\mathrm{E}-4$$

误差分别减小到原来的 0.52 和 0.53. 读者可对例 8.2.1(a) 和例 8.2.1(b) 中其他 x 值做相同的计算. 注意, 当 x 递增时, 误差的性态可能完全不同于相对误差的性态. 在第二个例子中, 当 x 递增时, 相对误差先递增后递减.

MATLAB 程序 下列 MATLAB 程序实施欧拉方法. 欧拉方法也称为向前欧拉方法. 在下节讨论向后欧拉方法.

```
function [x,y] = euler_for(x0,y0,x_end,h,fcn)
%
% function [x,y]=euler_for(x0,y0,x_end,h,fcn)
%
% Solve the initial value problem
%    y' = f(x,y),  x0 <= x <= b,  y(x0)=y0
% Use Euler's method with a stepsize of h.  The user must
% supply a program with some name, say, deriv, and a first
% line of the form
%   function ans=deriv(x,y)
% A sample call would be
%   [t,z]=euler_for(t0,z0,b,delta,'deriv')
%
% Output:
% The routine eulercls will return two vectors, x and y.
% The vector x will contain the node points
%   x(1)=x0, x(j)=x0+(j-1)*h, j=1,2,...,N
% with
%   x(N) <= x_end-h,  x(N)+h > x_end-h
% The vector y will contain the estimates of the solution Y
% at the node points in x.
%
```

```
n = fix((x_end-x0)/h)+1;
x = linspace(x0,x_end,n)';
y = zeros(n,1);
y(1) = y0;
for i = 2:n
  y(i) = y(i-1)+h*feval(fcn,x(i-1),y(i-1));
end
```

习题

1. 取步长 $h = 0.2, 0.1, 0.05$, 用欧拉方法求解下列问题. 利用真解 $Y(x)$ 计算误差和相对误差. 对选择的 x 值, 观察当 h 折半时这些误差减少的比率.

   (a) $Y'(x) = [\cos(Y(x))]^2, \quad 0 \leqslant x \leqslant 10, \quad Y(0) = 0;$

   $$Y(x) = \tan^{-1}(x)$$

   (b) $Y'(x) = \dfrac{1}{1+x^2} - 2[Y(x)]^2, \quad 0 \leqslant x \leqslant 10, \quad Y(0) = 0;$

   $$Y(x) = \frac{x}{1+x^2}$$

   (c) $Y'(x) = \dfrac{1}{4}Y(x)\left[1 - \dfrac{1}{20}Y(x)\right], \quad 0 \leqslant x \leqslant 20, \quad Y(0) = 1;$

   $$Y(x) = \frac{20}{1+19\mathrm{e}^{-x/4}}$$

   (d) $Y'(x) = -[Y(x)]^2, \quad 1 \leqslant x \leqslant 10, \quad Y(1) = 1; \quad Y(x) = \dfrac{1}{x}$

   (e) $Y'(x) = x\mathrm{e}^{-x} - Y(x), \quad 0 \leqslant x \leqslant 10, \quad Y(0) = 1;$

   $$Y(x) = \left(1 + \frac{1}{2}x^2\right)\mathrm{e}^{-x}$$

   (f) $Y'(x) = \dfrac{x^3}{Y(x)}, \quad 0 \leqslant x \leqslant 10, \quad Y(0) = 1; \quad Y(x) = \sqrt{\dfrac{1}{2}x^4 + 1}$

   (g) $Y'(x) = (3x^2+1)Y(x)^2, \quad 0 \leqslant x \leqslant 10, \quad Y(0) = -1;$

   $$Y(x) = -[x^3 + x + 1]^{-1}$$

2. 考虑 8.1 节 (8.3) 那样的线性方程

   $$Y'(x) = \lambda Y(x) + (1-\lambda)\cos(x) - (1+\lambda)\sin(x), \quad Y(0) = 1$$

   真解是 $Y(x) = \sin(x) + \cos(x)$, 对 $0 \leqslant x \leqslant 10$ 以及 λ 和 h 的几个值, 利用欧拉方法求解这个问题. 讨论所得的计算结果.

   (a) $\lambda = -1; \quad h = 0.5,\ 0.25,\ 0.125$

   (b) $\lambda = 1; \quad h = 0.5,\ 0.25,\ 0.125$

   (c) $\lambda = -5; \quad h = 0.5,\ 0.25,\ 0.125,\ 0.0625$

(d) $\lambda = 5;\quad h = 0.062\ 5$

3. 作为欧拉方法的误差可以直接分析的一个例子, 考虑将欧拉方法应用于问题

$$Y'(x) = Y(x), \quad Y(0) = 1$$

真解是 e^x.

(a) 证明欧拉方法的解可写成

$$y_h(x_n) = (1+h)^{x_n/h}, \quad n \geqslant 0$$

(b) 利用微积分里的洛比达法则, 证明

$$\lim_{h\to 0}(1+h)^{1/h} = e$$

于是, 对固定的 $x = x_n$,

$$\lim_{h\to 0} y_h(x) = e^x$$

(c) 我们来做更仔细的收敛性分析. 利用性质 $a^b = e^{b\ln a}$ 记

$$y_h(x_n) = e^{x_n \ln(1+h)/h}$$

然后利用公式

$$\ln(1+h) = h - \frac{h^2}{2} + O(h^3)$$

和自然指数函数的泰勒展开式证明

$$Y(x_n) - y_h(x_n) = -\frac{x_n e^{x_n}}{2}h + O(h^2)$$

这表明对小的 h, 误差几乎与 h 成比例. 从表 8-1 和表 8-2 中给出的数值结果我们已经观察到这个现象.

8.3 欧拉方法的收敛性分析

分析欧拉方法的目的是为了理解它; 并且能够预估它的误差, 因此加快数值解的收敛速度. 对欧拉方法的分析, 将帮助我们对其他更有效的数值方法研究类似的问题.

我们从直接导出欧拉方法的逼近 (8.22) 的误差入手. 利用泰勒定理有

$$Y(x_{n+1}) = Y(x_n) + hY'(x_n) + \frac{h^2}{2}Y''(\xi_n)$$

其中 ξ_n 满足 $x_n \leqslant \xi_n \leqslant x_{n+1}$. 由于 $Y(x)$ 满足微分方程, 所以可将上式变成

$$Y(x_{n+1}) = Y(x_n) + hf(x_n, Y(x_n)) + \frac{h^2}{2}Y''(\xi_n) \tag{8.27}$$

最后一项

$$T_{n+1}=\frac{h^2}{2}Y''(\xi_n) \tag{8.28}$$

称为欧拉方法的**截断误差**, 它是逼近

$$Y(x_{n+1})\approx Y(x_n)+hf(x_n,Y(x_n))$$

中的误差.

为分析欧拉方法的误差, 从 (8.27) 中减去

$$y_{n+1}=y_n+hf(x_n,y_n)$$

得到

$$Y(x_{n+1})-y_{n+1}=Y(x_n)-y_n+h[f(x_n,Y(x_n))-f(x_n,y_n)]+\frac{h^2}{2}Y''(\xi_n) \tag{8.29}$$

y_{n+1} 的误差分为两部分: (1) 截断误差 T_{n+1}, 这是在 x_{n+1} 步新引进的误差, (2) 传播误差

$$Y(x_n)-y_n+h\left[f(x_n,Y(x_n))-f(x_n,y_n)\right]$$

对 $f(x,z)$ 应用中值定理:

$$f(x_n,Y(x_n))-f(x_n,y_n)=\frac{\partial f(x_n,z_n)}{\partial z}[Y(x_n)-y_n]$$

其中 z_n 是位于 $Y(x_n)$ 和 y_n 之间的某个值. 记 $e_k\equiv Y(x_k)-y_k,\ k\geqslant 0$, 然后利用上式改写 (8.29) 为

$$e_{n+1}=\left[1+h\frac{\partial f(x_n,z_n)}{\partial z}\right]e_n+\frac{h^2}{2}Y''(\xi_n) \tag{8.30}$$

利用这些结果可给出初值问题欧拉方法的一般误差分析. 我们来考虑能够直观理解欧拉方法误差的一些特殊情况.

考虑利用欧拉方法求解问题

$$Y'(x)=2x,\quad Y(0)=0$$

它的真解是 $Y(x)=x^2$. 误差公式 (8.30) 变成

$$e_{n+1}=e_n+h^2,\quad e_0=0$$

由归纳法得,

$$e_n=nh^2,\quad n\geqslant 0$$

因为 $nh = x_n$, 所以

$$e_n = hx_n$$

对每个固定的 x_n, 在 x_n 处的误差和 h 成比例. 截断误差是 $O(h^2)$, 但这些误差累积的结果是一个与 h 成比例的整体误差.

利用 (8.30) 或 (8.29), 经过一些演算, 可以证明下列一般结果.

定理 8.3.1 假定在 $[x_0, b]$ 上真解 $Y(x)$ 二次连续可微, 并且

$$K = \sup_{\substack{-\infty < z < \infty \\ x_0 \leqslant x \leqslant b}} \left| \frac{\partial f(x,z)}{\partial z} \right| < \infty \tag{8.31}$$

则对一切 $x_0 \leqslant x_n \leqslant b$, 欧拉方法产生的数值解的误差满足不等式

$$\begin{aligned} |Y(x_n) - y_h(x_n)| \leqslant & \mathrm{e}^{(b-x_0)K} |Y_0 - y_0| \\ & + h \left[\frac{\mathrm{e}^{(b-x_0)K} - 1}{2K} \right] \max_{x_0 \leqslant x \leqslant b} |Y''(x)| \end{aligned} \tag{8.32}$$

证明可在许多高级数值分析教材中找到, 这里略去. 条件 (8.31) 使证明变得较容易, 但是我们可以像在定理 8.1.3 里那样弱化这个条件.

当 $y_0 = Y_0$ 时, (8.32) 可写成

$$|Y(x_n) - y_h(x_n)| \leqslant ch, \quad x_0 \leqslant x_n \leqslant b$$

其中 c 是一个常数. 这与前节表 8-1 和表 8-2 中观察到的误差性态相容, 并且它与定理 8.3.1 所考虑的特殊情况一致. 当 h 折半时, 界 ch 也折半, 这是前面已知的误差的性态. 欧拉方法称为 1 阶收敛的, 因为出现在误差界中 h 的幂是 1. 一般来说, 若有

$$|Y(x_n) - y_h(x_n)| \leqslant ch^p, \quad x_0 \leqslant x_n \leqslant b \tag{8.33}$$

则数值方法称为 p 阶收敛的. 自然地, 我们希望阶 p 越高, 收敛越快.

我们强调, 误差界 (8.32) 成立的一个条件是真解二次连续可微. 这个条件并不总是成立的. 当真解没有连续的二阶导数时, 误差界 (8.32) 不再成立, 见习题 6.

对一大类初值问题, 误差界 (8.32) 都成立. 但是, 由于误差界中出现指数项 h^p, 所以由这个误差界计算得到的具体数值常常是误差界上限. 在某些情况下, 我们可以改进结果. 假设偏导数

$$\frac{\partial f(x,z)}{\partial z} \leqslant 0$$

并且对一切 z 和在微分方程求解的区间 $[x_0, b]$ 内的一切 x, 这个偏导数的值有界. 注意这个假设与 8.1 节中稳定性条件 (8.19) 的关系. 再假设选择 h 使得 (8.30) 中

e_n 的系数满足

$$1+h\frac{\partial f(x,z)}{\partial z}\geqslant -1,\quad x_0\leqslant x\leqslant b,\quad -\infty<z<\infty$$

则由 (8.30) 可推出

$$|e_{n+1}|\leqslant |e_n|+ch^2,\quad x_0\leqslant x_n\leqslant b \tag{8.34}$$

其中

$$c=\frac{1}{2}\cdot\max_{x_0\leqslant x_n\leqslant b}|Y''(x)|$$

和 $e_0=0$. 归纳地应用 (8.34), 得到

$$|e_n|\leqslant nch^2=cx_nh \tag{8.35}$$

误差以一个与 h 成比例的量为界, 并且 h 项的系数与 x_n 成比例, 这与界 (8.32) 中系数的指数型增长大不相同.

定理 8.3.1 的误差估计是严格的, 它在洞察数值解的收敛性态中是有用的. 然而, 正如下例中指出的那样, 用 (8.32) 作为实际的误差界通常是不可取的.

例 8.3.2 考虑 8.2 节中已解决问题

$$Y'(x)=-Y(x),\quad Y(0)=1 \tag{8.36}$$

结果由表 8-1 给出. 为应用 (8.32), 我们有 $\frac{\partial f(x,z)}{\partial z}=-1$, $K=1$. 真解是 $Y(x)=\mathrm{e}^{-x}$, 因而

$$\max_{0\leqslant x\leqslant b}|Y''(x)|=1$$

并且 $y_0=Y_0=1$, 界 (8.32) 变成

$$\left|\mathrm{e}^{-x_n}-y_h(x_n)\right|\leqslant\frac{h}{2}(\mathrm{e}^b-1),\quad 0\leqslant x_n\leqslant b \tag{8.37}$$

当 $h\to 0$ 时, 上式表明 $y_h(x)$ 收敛于 e^{-x}. 然而, 这个界过分保守. 当 b 递增时, 这个界指数型递增. 对 $b=5$, 上面的界是

$$\left|\mathrm{e}^{-x_n}-y_h(x_n)\right|\leqslant\frac{h}{2}\left(\mathrm{e}^5-1\right)\approx 73.7h,\quad 0\leqslant x_n\leqslant 5$$

这个界比表 8-1 给出的实际误差大了好几个数量级. 对问题 (8.36), 改进的误差界 (8.35) 成立, 其中 $c=\dfrac{1}{2}$(参见习题 2). 精确地估计误差的一个更一般的方法在下一小节中讨论. ■

8.3.1 渐近误差分析

为了更准确地估计误差, 我们来考虑渐近误差展开, 类似于第 5 章对数值积分所做的分析. 在一定的条件下, 可以得到欧拉方法的渐近误差公式. 假定 Y 三次连续可微并且对靠近 $(x, Y(x))$ 的一切 (x, z) 值, $x_0 \leqslant x \leqslant b$, 两个偏导数

$$\frac{\partial f(x,z)}{\partial z}, \quad \frac{\partial^2 f(x,z)}{\partial z^2}$$

都连续. 则可以证明欧拉方法的误差满足

$$Y(x_n) - y_h(x_n) = hD(x_n) + O(h^2), \quad x_0 \leqslant x_n \leqslant b \tag{8.38}$$

通常情况下, 若假定 $y_0 = Y_0$, 函数 $D(x)$ 满足线性微分方程初值问题

$$D'(x) = g(x)D(x) + \frac{1}{2}Y''(x), \quad D(x_0) = 0 \tag{8.39}$$

其中

$$g(x) = \left.\frac{\partial f(x,z)}{\partial z}\right|_{z=Y(x)}$$

如果能求出函数 $D(x)$, 则公式 (8.38) 的主误差项 $hD(x_n)$ 通常是实际误差 $Y(x_n) - y_h(x_n)$ 的很好的估计, 并且估计量随着步长递减而越来越精确.

例 8.3.3 再考虑问题 (8.36). $D(x)$ 满足

$$D'(x) = -D(x) + \frac{1}{2}\mathrm{e}^{-x}, \quad D(0) = 0$$

它的解为

$$D(x) = \frac{1}{2}x\mathrm{e}^{-x}$$

当我们利用 (8.38) 时, 误差满足

$$Y(x_n) - y_h(x_n) \approx \frac{h}{2}x_n\mathrm{e}^{-x_n} \tag{8.40}$$

我们略去 $O(h^2)$ 项, 因为对一切充分小的 h 值, 它小于 (8.38) 中的项 $hD(x)$. 为检验 (8.40) 的精度, 考虑 $x_n = 5.0$ 和 $h = 0.05$. 于是估计的误差为

$$\frac{h}{2}x_n\mathrm{e}^{-x_n} \doteq 0.000\,842$$

由表 8-1 知, 实际的误差是 0.000 817, 十分接近于我们对它的估计. ■

8.3.2 理查森外推

试图从问题 (8.39) 来求出函数 $D(x)$ 是不实际的, 因为这需要知道真解; 并且即使可以得到 $Y''(x)$ 好的逼近, 求解问题 (8.39) 中的 $D(x)$ 代价也太高. 公式 (8.38) 真正的功能是精确地描述误差的性态. 在不知道函数 $D(x)$ 的情况下, 我们能利用 (8.38) 估计解的误差, 并改进数值解的精度. 为此, 我们需要两个数值解, 比如说 y_h 和 y_{2h}. 假如 x 是步长为 $2h$ 的节点. 则由公式 (8.38), 有

$$Y(x) - y_h(x) = hD(x) + O(h^2)$$
$$Y(x) - y_{2h}(x) = 2hD(x) + O(h^2)$$

从两个关系中消去 $D(x)$ 得到

$$Y(x) - [2y_h(x) - y_{2h}(x)] = O(h^2)$$

它也可写成

$$Y(x) - y_h(x) = y_h(x) - y_{2h}(x) + O(h^2)$$

丢弃高阶项 $O(h^2)$, 就得到*理查森外推公式*

$$Y(x) \approx \tilde{y}_h(x) \equiv 2y_h(x) - y_{2h}(x) \tag{8.41}$$

和*理查森误差估计*

$$Y(x) - y_h(x) \approx y_h(x) - y_{2h}(x) \tag{8.42}$$

用这些公式, 可以估计欧拉方法的误差并得到更迅速收敛的解 $\tilde{y}_h(x)$.

例 8.3.4 考虑用步长 $h = 0.05$, $2h = 0.1$ 解问题 (8.36). 表 8-3 给出了在一些结点处的理查森外推结果. 注意到 (8.42) 是误差相当准确的估计, 而 $\tilde{y}_h(x)$ 是比 $y_h(x)$ 更加精确的数值解. ■

表 8-3 带理查森外推的欧拉方法

x	$Y(x)-y_h(x)$	$y_h(x)-y_{2h}(x)$	$\tilde{y}_h(x)$	$Y(x)-\tilde{y}_h(x)$
1.0	9.39E−3	9.81E−3	3.682 934 6E−1	−4.14E−4
2.0	6.82E−3	6.94E−3	1.354 476 4E−1	−1.12E−4
3.0	3.72E−3	3.68E−3	4.974 844 3E−2	3.86E−5
4.0	1.80E−3	1.73E−3	1.824 987 7E−2	6.58E−5
5.0	8.17E−4	7.67E−4	6.687 285 3E−3	5.07E−5

利用 (8.38) 以及导出 (8.41) 的过程, 可以证明

$$Y(x_n) - \tilde{y}_h(x_n) = O(h^2) \tag{8.43}$$

这改进了欧拉方法的收敛速度. 我们将对后面引进的数值方法进行类似的外推. 当然, 对不同的数值方法, 外推公式和误差估计会不同于 (8.41) 和 (8.42), 具体的公式依赖于方法的阶.

习题

1. 对 $b=1,2,3,4,5$ 和 $h=0.2,0.1,0.05$, 检验误差界 (8.37) 的精度. 计算误差界并与表 8-1 中的结果比较.

2. 再次考虑例 8.3.2 的问题 (8.36). 我们来导出一个比定理 8.3.1 更精确的误差界. 由 (8.30), 有

$$e_{n+1}=(1-h)e_n+\frac{h^2}{2}\mathrm{e}^{-\xi_n}$$

利用这个公式及 $e_0=0$, 证明误差界

$$|e_n|\leqslant\frac{x_n}{2}h$$

把这个误差界与 8.2 节表 8-1 中的实际误差作比较.

3. 对 8.2 节的问题 (8.26), 假定 $y_0=Y_0$, 计算误差界 (8.32). 对 $b=1,2,3,4,5$ 和 $h=0.2,0.1,0.05$, 把误差界与表 8.2 中给出的实际误差作比较.

4. 对 8.2 节习题 1(a) 的方程, 重复习题 3.

5. 对 8.2 节习题 1 的 (b)~(d), (8.31) 中的常数 K 将是无穷大. 为了在这样的情况下使用误差界 (8.32), 取

$$K=2\cdot\max_{x_0\leqslant x_n\leqslant b}\left|\frac{\partial f(x,Y(x))}{\partial z}\right|$$

对所有充分小的 h 值, 可以证明这个值是恰当的. 对 8.2 节习题 1 的 (b)~(d), 重做上面的习题 3.

6. 考虑初值问题

$$Y'(x)=\alpha x^{\alpha-1},\quad Y(0)=0$$

其中 $\alpha>0$. 真解是 $Y(x)=x^\alpha$. 当 $\alpha\neq$ 整数时, 真解不是无限次可微的. 例如, 为使 Y 二次连续可微, 需要 $\alpha>2$. 对 $\alpha=2.5,1.5,1.1$ 和步长 $h=0.2,0.1,0.05$, 利用欧拉方法解初值问题. 计算节点上解的误差, 对这些问题数值上确定欧拉方法的收敛阶.

7. 初值问题

$$Y'(x)=\lambda Y(x)+\cos(x)-\lambda\sin(x),\quad Y(0)=0$$

的解是 $Y(x)=\sin(x)$. 求此情况的渐近误差公式 (8.38). 再对 $0\leqslant x\leqslant 6$, $h=0.2,0.1,0.05$ 和 $\lambda=1,-1$, 计算欧拉方法解. 把实际误差与从渐近估计

$$Y(x_n)-y_n\approx hD(x_n)$$

得到的那些值作比较.

8. 对 8.2 节的习题 1(d) 重复习题 7 并与 $1\leqslant x\leqslant 6$, $h=0.2,0.1,0.05$ 时的结果作比较.

9. 对 8.2 节的例 (8.26) 以及表 8-2 的数值结果, 当 $h=0.05$ 时, 利用理查森外推估计误差 $Y(x_n)-y_h(x_n)$. 另外, 产生理查森外推 $\tilde{y}_h(x_n)$ 并计算它的误差. 对 $x_n=1,2,3,4,5,6$ 做这项工作.

10. 对 8.2 节习题 1 的 (a) 到 (d), 重复习题 9.

11. 从 (8.38) 推导 (8.43).

8.4 数值稳定性, 隐式方法

8.1 节讨论了初值问题 (8.7) 的稳定性, 表明初始值小的扰动导致解小的改变. 对于求解初值问题的数值方法, 具有类似的稳定性是合乎需要的. 这意味着对任何充分小的步长 h, 初始值小的改变将导致数值解小的改变. 实际上, 这样的稳定性与收敛性密切相关. 对初值问题 (8.7) 在类似于定理 8.1.3 的那些假设之下, 一个有名的结果是具有二阶或更高阶截断误差的数值方法收敛当且仅当它是稳定的. 欧拉方法的截断误差 T_{n+1} 是 2 阶的 (参见 (8.28)). 定理 8.3.1 建立了方法的收敛性. 可以用类似于证明定理 8.3.1 的推理来证明方法的稳定性. 收敛性和稳定性之间关系的另一个例子是一个既不收敛又不稳定的数值方法, 参阅习题 13.

上面的稳定性表明, 倘若步长 h 充分小, 稳定的数值方法是良态的. 然而, 实际计算中, 步长 h 不可能太小, 因为非常小的步长降低数值方法的效率. 正如 5.4 节里曾讨论过的, 差分逼近, 例如用 $[Y(x+h)-Y(h)]/h$ 来逼近导数 $Y'(x)$, 当 h 过分小时其精度降低. 因此, 对实际计算来说, 重要的是当 h 不是很小时, 数值方法的性能怎样. 当不假定 h 很小时, 我们有必要进一步分析数值方法的稳定性.

对一般的问题

$$Y'(x)=f(x,Y(x)),\quad Y(x_0)=Y_0 \tag{8.44}$$

考察稳定性太复杂. 因此, 我们来对*模型方程*

$$Y'(x)=\lambda Y(x)+g(x),\quad Y(0)=Y_0 \tag{8.45}$$

考察数值方法的稳定性, 该方程的真解可从 (8.4) 求出. 对这个方程很容易回答所考虑的稳定性和收敛性问题, 并且可以证明对这个方程得到的结论往往对更一般的方程 (8.44) 也成立.

设 $Y(x)$ 是 (8.45) 的解, 并设 $Y_\varepsilon(x)$ 是对应于有扰动的初始数据 $Y_0+\varepsilon$ 的解,

$$Y_\varepsilon'(x)=\lambda Y_\varepsilon(x)+g(x),\quad Y_\varepsilon(0)=Y_0+\varepsilon$$

记 $Z_\varepsilon(x)$ 为两个解之差

$$Z_\varepsilon(x)=Y_\varepsilon(x)-Y(x)$$

接着, 从 $Y_\varepsilon(x)$ 的方程中减去 (8.45), 得到

$$Z_\varepsilon'(x)=\lambda Z_\varepsilon(x),\quad Z_\varepsilon(0)=\varepsilon$$

它的解为

$$Z_\varepsilon(x)=\varepsilon \mathrm{e}^{\lambda x}$$

在应用中, 我们一般对 $\lambda<0$ 或 λ 是具有负实部的复数情况感兴趣. 在这种情况, 当 $x\to\infty$ 时 $Z_\varepsilon(x)$ 将趋于零, 因而对大的 x 值, ε 扰动的影响消失. 我们希望应用于 (8.45) 的数值方法有类似的性质.

用函数 $Z_\varepsilon(x)/\varepsilon$ 代替 $Z_\varepsilon(x)$, 得到下列通常用于试验各种数值方法性能的模型问题:

$$\begin{cases} Y'=\lambda Y, & x>0 \\ Y(0)=1 \end{cases} \tag{8.46}$$

下面, 当提及模型问题 (8.46) 时, 总假定常数 $\lambda<0$ 或 λ 有负实部. (8.46) 的真解是

$$Y(x)=\mathrm{e}^{\lambda x} \tag{8.47}$$

因为参数 λ 有负实部, 真解随着 x 的增大而指数型地衰减.

我们想要的那种数值方法稳定性是当数值方法应用于 (8.46) 时, 对任何选取的步长 h, 数值解满足

$$y_h(x_n)\to 0, \quad 当\ x_n\to\infty\ 时 \tag{8.48}$$

这个性质称为**绝对稳定性**. 当 $h\to 0$ 时使 $y_n\to 0$ 的值 $h\lambda$ 的集合是复平面的一个子集, 称为数值方法的**绝对稳定性区域**.

我们来对模型问题 (8.46) 考察欧拉方法的性能. 我们有

$$y_{n+1}=y_n+h\lambda y_n=(1+h\lambda)y_n, \quad n\geqslant 0, \quad y_0=1$$

通过归纳论证, 不难求出

$$y_n=(1+h\lambda)^n, \quad n\geqslant 0 \tag{8.49}$$

注意, 对一个具有固定值的节点 $x_n=nh\equiv\bar{x}$, 当 $n\to\infty$ 时,

$$y_n=\left(1+\frac{\lambda\bar{x}}{n}\right)^n\to\mathrm{e}^{\lambda\bar{x}}$$

其中的极限可使用微积分中的洛比达法则得到. 这就证实了欧拉方法的收敛性. 这是当 $h\to 0$ 时成立的一个渐近性质.

从公式 (8.49) 看到, 当 $n\to\infty$ 时 $y_n\to 0$ 当且仅当

$$|1+h\lambda|<1$$

若 λ 是非负实数, 条件变成

$$h\lambda>-2 \tag{8.50}$$

这就决定应用欧拉方法时可以采用的 h 的范围: $0<h<(-2)/\lambda$.

例 8.4.1 对 $\lambda = -100$ 考虑模型问题. 则仅当 $h < 2 \times 100^{-1} = 0.02$ 时, 欧拉方法才可能提供有用的数值解. 在 $x = 0.2$ 处真解 $Y(x) = \mathrm{e}^{-100x}$ 的值是 2.061×10^{-9}. 对几个 h 值, 表 8-4 列出在 $x = 0.2$ 处的欧拉解. ■

表 8-4 对几个 h 值在 $x = 0.2$ 处的欧拉解

h	$y_h(0.2)$
0.1	81
0.05	256
0.02	1
0.01	0
0.001	7.06E−10

8.4.1 向后欧拉方法

现在考虑应用于模型问题 (8.46) 时, 对任意的步长 h 都具有性质 (8.48) 的数值方法. 这样的方法被称为绝对稳定的. 在欧拉方法的推导中, 使用的是向前差分逼近

$$Y'(x) \approx \frac{1}{h}[Y(x+h) - Y(x)]$$

如果使用向后差分逼近

$$Y'(x) \approx \frac{1}{h}[Y(x) - Y(x-h)]$$

则在 $x = x_n$ 处微分方程 $Y'(x) = f(x, Y(x))$ 的离散为

$$y_n = y_{n-1} + hf(x_n, y_n)$$

把指标位移 1 个单位, 就得到向后欧拉方法

$$\begin{cases} y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}), \quad 0 \leqslant n \leqslant N-1 \\ y_0 = Y_0 \end{cases} \tag{8.51}$$

像欧拉方法一样, 向后欧拉方法具有 1 阶精度, 而且类似于定理 8.3.1 的收敛性结果成立. (8.38) 形式的渐近误差展开式也有效.

下面证明向后欧拉方法应用于模型问题 (8.46) 时具有所要求的性质 (8.48). 我们有

$$y_{n+1} = y_n + h\lambda y_{n+1}$$
$$y_{n+1} = (1 - h\lambda)^{-1} y_n, \quad n \geqslant 0, \quad y_0 = 1$$

因此

$$y_n = (1 - h\lambda)^{-n} \tag{8.52}$$

对任意的步长 $h>0$, $|1-h\lambda|>0$, 所以当 $n\to\infty$ 时 $y_n\to 0$. 类似于例 8.4.1 中的数值结果, 我们在表 8-5 中给出向后欧拉方法相应的数值结果.

表 8-5　对几个 h 值在 $x=0.2$ 处的向后欧拉解

h	$y_h(0.2)$
0.1	8.26E−3
0.05	7.72E−4
0.02	1.69E−5
0.01	9.54E−7
0.001	5.27E−9

表 8-4 和表 8-5 之间的比较表明, 对模型问题 (8.46), 向后欧拉方法本质上比欧拉方法好.

在运行上, 两个方法之间的主要差别是, 在每个时间步向后欧拉方法需要求解一个关于 y_{n+1} 的非线性代数方程

$$y_{n+1}=y_n+hf(x_{n+1},y_{n+1}) \tag{8.53}$$

需要解代数方程才能求出 y_{n+1} 的方法称为*隐式方法*, 因为 y_{n+1} 是由代数方程隐式定义的. 与之相反, 直接给出 y_{n+1} 的方法称为*显式方法*. 欧拉方法是显式方法, 而向后欧拉方法是隐式方法. 在函数 $f(x,y)$ 关于 y 的利普希茨 (Lipschitz) 连续性假设之下, 应用定理 3.4.2 可以证明, 若 h 足够小, 则方程 (8.53) 有唯一解.

可对 (8.53) 应用第 3 章的求根方法求它的根 y_{n+1}, 但这样做通常非常耗时. 因此, 常用简单的迭代方法求解 (8.53). 给出初始猜测 $y_{n+1}^{(0)}\approx y_{n+1}$, 用

$$y_{n+1}^{(j+1)}=y_n+hf(x_{n+1},y_{n+1}^{(j)}),\quad j=0,1,2,\cdots \tag{8.54}$$

确定 $y_{n+1}^{(1)}$, $y_{n+1}^{(2)}$, 等等. 可以证明, 若 h 充分小, 则当 $j\to\infty$ 时迭代 $y_{n+1}^{(j)}$ 将收敛于 y_{n+1}. 从 (8.53) 减去 (8.54) 得到

$$y_{n+1}-y_{n+1}^{(j+1)}=h[f(x_{n+1},y_{n+1})-f(x_{n+1},y_{n+1}^{(j)})]$$

$$y_{n+1}-y_{n+1}^{(j+1)}\approx h\cdot\frac{\partial f(x_{n+1},y_{n+1})}{\partial z}[y_{n+1}-y_{n+1}^{(j)}]$$

把 $f(x_{n+1},z)$ 看作是 z 的函数, 对 $f(x_{n+1},z)$ 应用中值定理得到最后的公式. 这个公式给出逐次迭代误差之间的关系. 所以, 只要初始猜测 $y_{n+1}^{(0)}$ 足够好, 若

$$\left|h\cdot\frac{\partial f(x_{n+1},y_{n+1})}{\partial z}\right|<1 \tag{8.55}$$

则误差将收敛于零.

上面的迭代方法 (8.54) 及其分析仅是 3.4 节不动点迭代理论的特殊情况. 按前面材料的记号, $\alpha = y_{n+1}$ 是不动点, 并且

$$g(z) \equiv y_n + hf(x_{n+1}, z)$$

收敛条件 (8.55) 就是前面 (3.47) 的条件 $|g'(\alpha)| < 1$.

实际使用时, 可利用一个好的初始猜测, 并且选择 h 使 (8.55)左边的值比 1 小得多. 则误差 $\left|y_{n+1} - y_{n+1}^{(j)}\right|$ 会迅速减小, 并且通常只需要计算一次迭代. 作为 (8.54) 的初始猜测 $y_{n+1}^{(0)}$, 我们可利用欧拉方法

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n)$$

它称为**预估公式**, 因为它预估隐式方法的根.

对许多方程, 通常做一次迭代 (8.54) 就足够了. 因而实施向后欧拉方法的实际方法是做下列近似地求解 (8.53) 的单点迭代:

$$\begin{aligned} \bar{y}_{n+1} &= y_n + hf(x_{n+1}, y_n) \\ y_{n+1} &= y_n + hf(x_{n+1}, \bar{y}_{n+1}) \end{aligned}$$

最终得到的数值方法由公式

$$y_{n+1} = y_n + hf(x_{n+1}, y_n + hf(x_{n+1}, y_n)) \tag{8.56}$$

表达. 可以证明这个方法仍具有一阶精度. 然而, 它不再是绝对稳定的 (参见习题 1).

MATLAB 程序 现在转到向后欧拉方法的实施. 在每步, 基于前一步得到的 y_n, 由欧拉方法计算 y_{n+1} 的估计值:

$$y_{n+1}^{(1)} = y_n + hf(x_n, y_n)$$

然后使用向后欧拉公式做迭代

$$y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)})$$

一直到迭代的相继值之间的差充分小, 从而得到解 y_{n+1} 的一个足够准确的近似值. 为防止迭代的无限循环, 如果作 10 次迭代还未得到满意的解, 就停止迭代; 在后面这种情况, 将显示一个错误信息.

```
function [x,y] = euler_back(x0,y0,x_end,h,fcn,tol)
%
% function [x,y] = euler_back(x0,y0,x_end,h,fcn,tol)
%
```

```
% Solve the initial value problem
%    y' = f(x,y),  x0 <= x <= b,  y(x0)=y0
% Use the backward Euler method with a stepsize of h.
% The user must supply an m-file to define the
% derivative f, with some name, say, 'deriv.m', and a
% first line of the form
%     function ans=deriv(x,y)
% tol is the user supplied bound on the difference
% between successive values of the trapezoidal
% iteration.  A sample call would be
%   [t,z]=euler_back(t0,z0,b,delta,'deriv',1.0e-3)
%
% Output:
% The routine euler_back will return two vectors,
% x and y. The vector x will contain the node points
%   x(1)=x0, x(j)=x0+(j-1)*h, j=1,2,...,N
% with
%   x(N) <= x_end,  x(N)+h > x_end
% The vector y will contain the estimates of the
% solution Y at the node points in x.
%

% Initialize.
n = fix((x_end-x0)/h)+1;
x = linspace(x0,x_end,n)';
y = zeros(n,1);
y(1) = y0;
i = 2;
% advancing
while i <= n
%
% forward Euler estimate
%
  yt1 = y(i-1)+h*feval(fcn,x(i-1),y(i-1));
% one-point iteration
  count = 0;
  diff = 1;
  while diff > tol & count < 10
    yt2 = y(i-1) + h*feval(fcn,x(i),yt1);
    diff = abs(yt2-yt1);
    yt1 = yt2;
    count = count +1;
  end
  if count >= 10
    disp('Not converging after 10 steps at x = ')
```

```
    fprintf('%5.2f\n', x(i))
  end
  y(i) = yt2;
  i = i+1;
end
```

8.4.2 梯形方法

欧拉方法和向后欧拉方法的主要缺点是收敛阶低. 下面给出一个具有较高收敛阶的方法, 而且用于求解模型问题 (8.46) 时, 对任意的步长 h, 稳定性性质 (8.48) 成立.

对微分方程

$$Y'(x) = f(x, Y(x))$$

两边从 x_n 到 x_{n+1} 积分得到

$$Y(x_{n+1}) = Y(x_n) + \int_{x_n}^{x_{n+1}} f(x, Y(x))\mathrm{d}x$$

然后利用梯形法近似积分

$$Y(x_{n+1}) \approx Y(x_n) + \frac{h}{2}\left[f(x_n, Y(x_n)) + f(x_{n+1}, Y(x_{n+1}))\right]$$

让两边相等, 得到求解初值问题 (8.7) 的梯形方法:

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right], \quad n \geqslant 0 \\ y_0 = Y_0 \end{cases} \tag{8.57}$$

可以证明梯形方法具有 2 阶精度, 并且它是绝对稳定的.

注意到梯形方法是一个隐式方法. 在一般步中, 需要从方程

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right] \tag{8.58}$$

中求出 y_{n+1}, 然而仅在少数情况可以直接求解它. 可将有关向后欧拉方程 (8.53) 的解的讨论稍作变化, 然后应用到解方程 (8.58). 用

$$y_{n+1}^{(j+1)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(j)})\right], \quad j = 0, 1, 2, \cdots \tag{8.59}$$

替代迭代公式 (8.54). 如果 h 充分小, 则当 $j \to \infty$ 时, 迭代 $y_{n+1}^{(j)}$ 将收敛到 y_{n+1}. 收敛性条件为

$$\left|\frac{h}{2} \cdot \frac{\partial f(x_{n+1}, y_{n+1})}{\partial z}\right| < 1 \tag{8.60}$$

注意条件 (8.60) 比 (8.55) 更容易满足, 表明梯形方法比向后欧拉方法更易于使用.

通常选择 (8.59) 的初始猜测是基于欧拉方法

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n) \tag{8.61}$$

或二阶亚当斯–巴什福思 (Adams-Bashforth) 方法 (参见 8.6 节)

$$y_{n+1}^{(0)} = y_n + \frac{h}{2}[3f(x_n, y_n) - f(x_{n-1}, y_{n-1})] \tag{8.62}$$

这些方法称为**预估公式**. 对上面两种情况的任一种, 从 (8.59) 计算 $y_{n+1}^{(1)}$ 并把它作为根 y_{n+1}. 对于选择 $y_{n+1}^{(0)}$ 的两个方法, 可以证明产生的解 $\{y_h(x_n)\}$ 的整体误差仍是 $O(h^2)$. 如果使用欧拉预估公式 (8.61) 确定 $y_{n+1}^{(0)}$, 并采用 $y_{n+1}^{(1)}$ 作为 y_{n+1} 的值, 则得到的新格式是

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n))] \tag{8.63}$$

这个方法称为**海恩 (Heun) 方法**. 海恩方法仍具有 2 阶精度. 然而, 它不再绝对稳定.

MATLAB 程序 在实施梯形方法的每一步, 基于前一步得到的 y_n, 我们使用欧拉方法计算 y_{n+1} 的估计值:

$$y_{n+1}^{(1)} = y_n + hf(x_n, y_n)$$

然后使用梯形公式做迭代

$$y_{n+1}^{(k+1)} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(k)})\right]$$

直到迭代的相继值之间的差充分小, 从而得到解 y_{n+1} 的一个足够精确的近似值. 为防止迭代无限循环, 如果作 10 次迭代还未得到满意的解, 则停止迭代; 在后面这种情况, 将显示一个错误信息.

```
function [x,y] = trapezoidal(x0,y0,x_end,h,fcn,tol)
%
% function [x,y] = trapezoidal(x0,y0,x_end,h,fcn,tol)
%
% Solve the initial value problem
%    y' = f(x,y),  x0 <= x <= b,  y(x0)=y0
% Use trapezoidal method with a stepsize of h.  The
% user must supply an m-file to define the derivative
% f, with some name, say, 'deriv.m', and a first line
% of the form
%   function ans=deriv(x,y)
```

```
% tol is the user supplied bound on the difference
% between successive values of the trapezoidal
% iteration. A sample call would be
%   [t,z]=trapezoidal(t0,z0,b,delta,'deriv',1e-3)
%
% Output:
% The routine trapezoidal will return two vectors,
% x and y. The vector x will contain the node points
%   x(1) = x0, x(j) = x0+(j-1)*h, j=1,2,...,N
% with

%   x(N) <= x_end,  x(N)+h > x_end
% The vector y will contain the estimates of the
% solution Y at the node points in x.
%

% Initialize.
n = fix((x_end-x0)/h)+1;
x = linspace(x0,x_end,n)';
y = zeros(n,1);
y(1) = y0;
i = 2;
% advancing
while i <= n
  fyt = feval(fcn,x(i-1),y(i-1));
%
% Euler estimate
%
  yt1 = y(i-1)+h*fyt;
% trapezoidal iteration
  count = 0;
  diff = 1;
  while diff > tol & count < 10
    yt2 = y(i-1) + h*(fyt+feval(fcn,x(i),yt1))/2;
    diff = abs(yt2-yt1);
    yt1 = yt2;
    count = count +1;
  end
  if count >= 10
    disp('Not converging after 10 steps at x = ')
    fprintf('%5.2f\n', x(i))
  end
  y(i) = yt2;
  i = i+1;
end
```

例 8.4.2 考虑问题

$$Y'(x) = \lambda Y(x) + (1-\lambda)\cos(x) - (1+\lambda)\sin(x), \quad Y(0) = 1 \tag{8.64}$$

其真解是 $Y(x) = \sin(x) + \cos(x)$. 对几个 λ 和 h 的值, 用欧拉方法求数值解, 计算结果在表 8-6 中给出. 注意, 按照公式 (8.28), 截断误差为

$$T_{n+1} = \frac{h^2}{2} Y''(\xi_n)$$

表 8-6 (8.64) 的欧拉方法

λ	x	误差: $h=0.5$	误差: $h=0.1$	误差: $h=0.01$
−1	1	−2.46E−1	−4.32E−2	−4.22E−3
	2	−2.55E−1	−4.64E−2	−4.55E−3
	3	−2.66E−2	−6.78E−3	−7.22E−4
	4	2.27E−1	3.91E−2	3.78E−3
	5	2.72E−1	4.91E−2	4.81E−3
−10	1	3.98E−1	−6.99E−3	−6.99E−4
	2	6.90E+0	−2.90E−3	−3.08E−4
	3	1.11E+2	3.86E−3	3.64E−4
	4	1.77E+3	7.07E−3	7.04E−4
	5	2.83E+4	3.78E−3	3.97E−4
−50	1	3.26E+0	1.06E+3	−1.39E−4
	2	1.88E+3	1.11E+9	−5.16E−5
	3	1.08E+6	1.17E+15	8.25E−5
	4	6.24E+8	1.23E+21	1.41E−4
	5	3.59E+11	1.28E+27	7.00E−5

因为 $Y(x)$ 与 λ 无关, 所以截断误差与 λ 无关. 但是实际的整体误差强烈地依赖于 λ, 如表中显示的那样; 整体误差的性态与 λh 的大小直接相关, 因而与欧拉方法稳定性区域的大小直接相关. 倘若 λh 足够小, 则误差就小. 不稳定且误差快速增大的情况正好是 λh 在范围 (8.50) 之外的情况.

现在用向后欧拉方法和梯形方法来求问题 (8.64) 的解. 表 8-7 和表 8-8 中给出步长 $h = 0.5$ 时的结果. 误差随 λ 而变化, 但和欧拉方法大不相同的是, 不再存在稳定性问题. 向后欧拉方法和梯形方法的解 y_{n+1} 是精确地计算的. 因为微分方程关于 Y 是线性的, 所以这是可能的. 当 $|\lambda h|$ 大时, 不动点迭代 (8.54) 和 (8.59) 不收敛. ■

表 8-7 (8.64) 的向后欧拉法解, $h = 0.5$

x	误差: $\lambda=-1$	误差: $\lambda=-10$	误差: $\lambda=-50$
2	2.08E−1	1.97E−2	3.60E−3
4	−1.63E−1	−3.35E−2	−6.94E−3
6	−7.04E−2	8.19E−3	2.18E−3
8	2.22E−1	2.67E−2	5.13E−3
10	−1.14E−1	−3.04E−2	−6.45E−3

表 8-8 (8.64) 的向后梯形法解, $h = 0.5$

x	误差: $\lambda=-1$	误差: $\lambda=-10$	误差: $\lambda=-50$
2	−1.13E−2	−2.78E−3	−7.91E−4
4	−1.43E−2	−8.91E−5	−8.91E−5
6	2.02E−2	2.77E−3	4.72E−4
8	−2.86E−3	−2.22E−3	−5.11E−4
10	−1.79E−2	−9.23E−4	−1.56E−4

形如 (8.45) 的方程

$$Y'(x) = \lambda Y(x) + g(x)$$

当 $\lambda < 0$ 且 $|\lambda|$ 很大时是所谓的*刚性* (stiff) *微分方程*. 刚性微分方程的特征是: 当它用不是绝对稳定的数值格式求解时, 为了能得到合理的近似解通常需要强迫步长 h 过分地小. 参见例 8.4.1 和例 8.4.2 利用欧拉方法解模型方程 (8.45) 和 (8.64) 中关于限制 h 大小的讨论. 对刚性微分方程, 必须使用绝对稳定的数值方法或有大的绝对稳定域的数值方法. 通常求解刚性微分方程时需要用隐式方法而不是显式方法. MATLAB 内置函数`ode23tb`是为解刚性微分方程特别设计的. 它的用法类似于 8.5.3 节中说明的另一个 MATLAB 内置函数`ode45`.

习题

1. 证明由公式 (8.56) 定义的方法不是绝对稳定的.
2. 证明梯形方法 (8.57) 是绝对稳定的, 但格式 (8.63) 不是绝对稳定的.
3. 利用向后欧拉方法解 8.2 节的习题 2.
4. 利用梯形法解 8.2 节的习题 2.
5. 对 $\alpha = 2.5, 1.5, 1.1$ 及 $h = 0.2, 0.1, 0.05$, 应用向后欧拉方法解 8.3 节习题 6 中的初值问题. 计算节点处解的误差, 确定数值收敛阶, 并把计算结果与欧拉方法的结果作比较.
6. 对 $\alpha = 2.5, 1.5, 1.1$ 及 $h = 0.2, 0.1, 0.05$, 应用梯形方法解 8.3 节习题 6 的初值问题. 计算在节点处解的误差, 确定数值收敛阶, 并且把计算结果与欧拉方法和向后欧拉方法的结果作比较.
7. 解方程

$$Y'(x) = \lambda Y(x) + \frac{1}{1+x^2} - \lambda \tan^{-1}(x), \quad Y(0) = 0$$

真解为 $Y(x) = \tan^{-1}(x)$. 分别使用欧拉方法、向后欧拉方法和梯形法求数值解. 设 $\lambda = -1, -10, -50$ 和 $h = 0.5, 0.1, 0.001$. 讨论所得到的计算结果.

8. 应用向后欧拉方法数值求解 $Y'(x) = \lambda Y(x) + g(x)$, $\lambda < 0$ 且 $|\lambda|$ 很大. 为了使迭代

$$y_{n+1}^{(j+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(j)}), \quad j = 0, 1, 2 \cdots$$

收敛于 y_{n+1}, 必须选择多小的 h? 对大的 $|\lambda|$ 值, 这个迭代法实用吗?

9. 确定中点方法

$$y_{n+1} = y_n + hf\left(x_{n+1/2}, \frac{y_n + y_{n+1}}{2}\right)$$

是否绝对稳定, 这里 $x_{n+1/2} = (x_n + x_{n+1})/2$.

10. 设 $\theta \in [0,1]$ 是一个常数, $x_{n+\theta} = (1-\theta)x_n + \theta x_{n+1}$. 考虑广义的中点法

$$y_{n+1} = y_n + hf(x_{n+\theta}, (1-\theta)y_n + \theta y_{n+1})$$

和它的梯形模拟方法

$$y_{n+1} = y_n + h\left[(1-\theta)f(x_n, y_n) + \theta f(x_{n+1}, y_{n+1})\right]$$

证明当 $\theta \in [1/2, 1]$ 时, 方法绝对稳定. 当 $0 \leqslant \theta < 1/2$ 时, 确定方法的绝对稳定性区域.

11. 作为向后欧拉方法的误差能被直接分析的特殊情况, 我们再次考虑模型问题 (8.46), 其中 λ 是任意的实常数. 问题的向后欧拉解由公式 (8.52) 给出. 类似于 8.2 节中的习题 3(c), 证明

$$Y(x_n) - y_h(x_n) = -\frac{\lambda^2 x_n \mathrm{e}^{\lambda x_n}}{2} h + O(h^2)$$

12. 同习题 11, 考虑带实常数 λ 的模型问题 (8.46). 证明梯形方法的解是

$$y_h(x_n) = \left(\frac{1+\lambda h/2}{1-\lambda h/2}\right)^n, \quad n \geqslant 0$$

把解的公式改写为

$$y_h(x_n) = \exp \frac{x_n\left[\ln(1+\lambda h/2) - \ln(1-\lambda h/2)\right]}{h}$$

再利用泰勒展开式证明

$$Y(x_n) - y_h(x_n) = -\frac{\lambda^3 x_n \mathrm{e}^{\lambda x_n}}{12} h^2 + O(h^4)$$

故对小的 h, 误差几乎是与 h^2 成比例.

13. 在本练习中, 考虑一个具有 3 阶截断误差的方法, 这个方法既不收敛也不稳定.

(a) 若 $Y(x)$ 是三次连续可微的, 证明

$$\begin{aligned} Y(x_{n+1}) =& 3Y(x_n) - 2Y(x_{n-1}) + \frac{h}{2}\left[Y'(x_n) - 3Y'(x_{n-1})\right] \\ &+ \frac{7}{12}h^3 Y'''(x_n) + O(h^4) \end{aligned} \tag{8.65}$$

因而, 求解微分方程

$$Y'(x) = f(x, Y(x))$$

的一个数值方法是

$$y_{n+1} = 3y_n - 2y_{n-1} + \frac{h}{2}\left[f(x_n, y_n) - 3f(x_{n-1}, y_{n-1})\right], \quad n \geqslant 1$$

这是截断误差为 $O(h^3)$ 的一个数值方法. 它是多步法 (参见 8.6 节) 的例子. 为了使用这个方法, 除初值 $y_0 = Y_0$ 外还需要一个人为的初值 y_1.

提示: 为证明 (8.65), 利用关于点 x_n 的泰勒展开式.

(b) 现在应用这个方法求解一个非常简单的初值问题

$$Y'(x) \equiv 0, \quad Y(0) = 1$$

问题的真解是 $Y(x) \equiv 1$. 证明：若初值选择 $y_0 = 1$, $y_1 = 1 + h$, 则数值解是 $y_n = 1 - h + h2^n$. 注意, 当 $h \to 0$ 时, $|y_1 - Y(h)| = h \to 0$. 设 $x_n = 1$. 证明：当 $h \to 0$ 时, $|Y(1) - y_n| \to \infty$. 因而方法不收敛.

(c) 把 (b) 的论证稍作变形可用于证明方法的不稳定性. 对初值 $y_0 = y_1 = 1$, 证明对一切 n, 数值解是 $y_n = 1$, 然而当初值扰动为 $y_{\varepsilon,0} = 1$, $y_{\varepsilon,1} = 1 + \varepsilon$ 时, 数值解变成 $y_{\varepsilon,n} = 1 - \varepsilon + \varepsilon 2^n$. 证明：在任何固定节点 $x_n = \bar{x} > 0$ 处, 当 $h \to 0$ 时, $|y_{\varepsilon,n} - y_n| \to \infty$. 因此方法不稳定.

8.5 泰勒方法和龙格–库塔方法

提高欧拉方法收敛速度的一个途径是寻找逼近 $Y(x_{n+1})$ 的更精确的公式, 欧拉方法利用了下述逼近

$$Y(x_{n+1}) \approx Y(x_n) + hY'(x_n)$$

因为这是线性泰勒逼近, 所以很自然地进一步考虑高阶泰勒逼近. 这样做将导出一族方法, 它与所使用的泰勒逼近的阶有关.

为使方法的描述尽可能地直观, 我们将对问题

$$Y'(x) = -Y(x) + 2\cos(x), \quad Y(0) = 1 \tag{8.66}$$

讨论泰勒方法. 问题的真解是 $Y(x) = \sin(x) + \cos(x)$. 为了利用在 x_n 处与 Y 有关的信息逼近 $Y(x_{n+1})$, 利用二次泰勒逼近

$$Y(x_{n+1}) \approx Y(x_n) + hY'(x_n) + \frac{h^2}{2}Y''(x_n) \tag{8.67}$$

它的截断误差是

$$T_{n+1} = \frac{h^3}{6}Y'''(\xi_n), \quad \text{其中 } x_n \leqslant \xi_n \leqslant x_{n+1} \tag{8.68}$$

为了计算 (8.67) 右边的值, 可以从 (8.66) 直接得到 $Y'(x_n)$. 对 (8.66) 求导得到

$$Y''(x) = -Y'(x) - 2\sin(x) = Y(x) - 2\cos(x) - 2\sin(x)$$

于是 (8.67) 变成

$$\begin{aligned} Y(x_{n+1}) \approx{}& Y(x_n) + h\left[-Y(x_n) + 2\cos(x_n)\right] \\ &+ \frac{h^2}{2}\left[Y(x_n) - 2\cos(x_n) - 2\sin(x_n)\right] \end{aligned}$$

让上式两边相等, 就得到数值方法

$$\begin{aligned} y_{n+1} = & y_n + h[-y_n + 2\cos(x_n)] \\ & + \frac{h^2}{2}[y_n - 2\cos(x_n) - 2\sin(x_n)], \quad n \geqslant 0 \end{aligned} \tag{8.69}$$

其中 $y_0 = 1$. 这个方法会逼近问题 (8.66) 的解. 因为截断误差 (8.68) 中 h 的幂次比欧拉方法的截断误差 [见 (8.28)] 中 h 的幂次更高, 所以我们期望方法 (8.69) 会比欧拉方法收敛得快.

表 8-9 给出了方法 (8.69) 的数值结果以及欧拉方法的数值结果, 显然 (8.69) 较好. 此外, 若比较相应于步长 $h = 0.1$ 和 $h = 0.05$ 的数值结果, 可以看出当 h 折半时误差大约减少到原来的 1/4. 后面将从理论上证明这个结果.

表 8-9 二阶泰勒方法 (8.69) 的例

h	x	$y_h(x)$	误差	欧拉误差
0.1	2.0	0.492 225 829	9.25E−4	−4.64E−2
	4.0	−1.411 659 477	1.21E−3	3.91E−2
	6.0	0.682 420 081	−1.67E−3	1.39E−2
	8.0	0.843 648 978	2.09E−4	−5.07E−2
	10.0	−1.384 588 757	1.50E−3	2.83E−2
0.05	2.0	0.492 919 943	2.31E−4	−2.30E−2
	4.0	−1.410 737 402	2.91E−4	1.92E−2
	6.0	0.681 162 413	−4.08E−4	6.97E−3
	8.0	0.843 801 368	5.68E−5	−2.50E−2
	10.0	−1.383 454 154	3.62E−4	1.39E−2

一般地, 求解初值问题

$$Y'(x) = f(x, Y(x)), \quad x_0 \leqslant x \leqslant b, \quad Y(x_0) = Y_0 \tag{8.70}$$

我们选择一个泰勒逼近的阶并作类似于上面那样的处理. 设逼近阶为 p, 记

$$Y(x_{n+1}) \approx Y(x_n) + hY'(x_n) + \cdots + \frac{h^p}{p!}Y^{(p)}(x_n) \tag{8.71}$$

其截断误差是

$$T_{n+1} = \frac{h^{p+1}}{(p+1)!}Y^{(p+1)}(\xi_n), \quad x_n \leqslant \xi_n \leqslant x_{n+1} \tag{8.72}$$

通过对 (8.70) 中的微分方程逐次求导可得 $Y''(x), \cdots, Y^{(p)}(x)$, 只涉及 x_n 和 $Y(x_n)$ 的表达式. 例如, 我们有下列公式

$$Y''(x) = f_x + f_z f$$

$$Y^{(3)}(x) = f_{xx} + 2f_{xz}f + f_{zz}f^2 + f_z(f_x + f_z f)$$

其中

$$f_x = \frac{\partial f}{\partial x}, \quad f_z = \frac{\partial f}{\partial z}, \quad f_{xz} = \frac{\partial^2 f}{\partial x \partial z}$$

是偏导数, 它们和 f 都在 $(x, Y(x))$ 处求值. 当微分阶数增加时, 高阶导数的公式变得非常复杂.

把这些公式代入到 (8.71) 中, 使 (8.71) 为等式, 就得到形如

$$y_{n+1} = y_n + hy_n' + \frac{h^2}{2}y_n'' + \cdots + \frac{h^p}{p!}y_n^{(p)} \tag{8.73}$$

的数值方法.

若解 $Y(x)$ 和函数 $f(x,z)$ 充分可微, 则可以证明方法 (8.73) 将满足

$$\max_{x_0 \leqslant x_n \leqslant b} |Y(x_n) - y_h(x_n)| \leqslant ch^p \cdot \max_{x_0 \leqslant x \leqslant b} |Y^{(p+1)}(x)| \tag{8.74}$$

其中常数 c 与欧拉方法的误差公式 (8.32) 中出现常数的相类似. 此外, 存在渐近误差公式

$$Y(x_n) - y_h(x_n) = h^p D(x_n) + O(h^{p+1}) \tag{8.75}$$

其中 $D(x)$ 满足某个线性微分方程. 误差公式 (8.74) 表明, 利用泰勒逼近可得到收敛阶数与之相同的数值方法. 误差渐近展开式 (8.75) 容许我们进行理查森外推以估计误差和加速收敛.

例 8.5.1 对 $p = 2$, 公式 (8.75) 导致

$$Y(x_n) - y_h(x_n) \approx \frac{1}{3}[y_h(x_n) - y_{2h}(x_n)] \tag{8.76}$$

它的推导作为习题 3 留给读者. 为说明这个公式作用, 利用表 8-9 中相应于 $x_n = 10$ 的数值解.

$$y_{0.1}(10) = -1.384\ 588\ 757$$
$$y_{0.05}(10) = -1.383\ 454\ 154$$

根据 (8.76)

$$Y(10) - y_{0.05}(10) \doteq \frac{1}{3}[0.001\ 134\ 603] \doteq 3.78\text{E} - 4$$

这是表 8-9 给出的实际误差 3.62E − 4 的一个令人满意的估计. ■

8.5.1 龙格–库塔方法

泰勒方法理论上容易操作, 但正如我们所看到的那样, 计算高阶导数是冗长和费时的. 为避免使用高阶导数, 龙格–库塔方法通过在多个点上计算 $f(x,z)$ 的值来达到高精度. 所得到的方法容易编程, 是求解初值问题最流行的方法.

首先考虑二阶龙格–库塔方法, 后面会讨论一些较高阶的方法. 龙格–库塔方法的一般形式为

$$y_{n+1} = y_n + hF(x_n, y_n; h), \quad n \geqslant 0, \quad y_0 = Y_0 \tag{8.77}$$

$F(x_n, y_n; h)$ 可看作区间 $[x_n, x_{n+1}]$ 上解的某种 “平均斜率”. 但是它的构造是基于使 (8.77) 像泰勒方法一样. 关于二阶方法, 一般选择

$$F(x, y; h) = \gamma_1 f(x, y) + \gamma_2 f(x + \alpha h, y + \beta h f(x, y)) \tag{8.78}$$

并确定常数 $\{\alpha, \beta, \gamma_1, \gamma_2\}$ 使得当真解代入到 (8.77) 中时, 如二阶泰勒方法那样, 截断误差

$$T_{n+1} \equiv Y(x_{n+1}) - [Y(x_n) + hF(x_n, Y(x_n); h)] \tag{8.79}$$

将是 $O(h^3)$.

为找到这些常数应满足的方程, 我们用泰勒展开式去计算截断误差 T_{n+1}. 对 $f(x+\alpha h, y+\beta h f(x,y))$ 项, 首先关于第二个自变量在 y 附近展开. 注意到我们需要形如余项 $O(h^2)$:

$$f(x+\alpha h, y+\beta h f(x,y)) = f(x+\alpha h, y) + f_z(x+\alpha h, y)\beta h f(x,y) + O(h^2)$$

然后, 关于第一个自变量在 x 附近展开, 得到

$$f(x+\alpha h, y+\beta h f(x,y)) = f + f_x \alpha h + f_z \beta h f + O(h^2)$$

其中所有的函数都在 (x, y) 处求值. 由微分方程得到

$$Y'' = f_x + f_z f$$

这样,

$$\begin{aligned} Y(x+h) &= Y + hY' + \frac{h^2}{2}Y'' + O(h^3) \\ &= Y + hf + \frac{h^2}{2}(f_x + f_z f) + O(h^3) \end{aligned}$$

于是

$$\begin{aligned} &Y(x+h) - [Y(x) + hF(x, Y(x); h)] \\ =&\, Y + hf + \frac{h^2}{2}(f_x + f_z f) - [Y + h\gamma_1 f + \gamma_2 h(f + \alpha h f_x + \beta h f_z f)] + O(h^3) \\ =&\, h(1-\gamma_1-\gamma_2)f + \frac{h^2}{2}[(1-2\gamma_2\alpha)f_x + (1-2\gamma_2\beta)f_z f] + O(h^3) \end{aligned}$$

故系数必须满足方程组

$$\begin{cases} 1-\gamma_1-\gamma_2=0 \\ 1-2\gamma_2\alpha=0 \\ 1-2\gamma_2\beta=0 \end{cases}$$

所以

$$\gamma_2 \neq 0, \quad \gamma_1 = 1-\gamma_2, \quad \alpha=\beta=\frac{1}{2\gamma_2} \tag{8.80}$$

因而, 存在一组二阶龙格–库塔方法, 这些方法与 γ_2 的选择有关. 三种最常见的选择是 $\gamma_2=\frac{1}{2},\frac{3}{4},1$. 取 $\gamma_2=\frac{1}{2}$, 得到数值方法

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_n+h,y_n+hf(x_n,y_n))], \quad n\geqslant 0 \tag{8.81}$$

这也是 8.4 节中讨论过的海恩方法. $y_n+hf(x_n,y_n)$ 是在 x_{n+1} 处的欧拉解. 利用它, 得到在 x_{n+1} 处的导数的近似值, 即

$$f(x_{n+1},y_n+hf(x_n,y_n))$$

然后把这个值和斜率 $f(x_n,y_n)$ 平均, 给出在区间 $[x_n,x_{n+1}]$ 上解的 "平均" 斜率, 得到

$$F(x_n,y_n;h)=\frac{1}{2}\left[f(x_n,y_n)+f(x_n+h,y_n+hf(x_n,y_n))\right]$$

然后在 (8.81) 中使用这个值, 从 y_n 预估 y_{n+1}. 图 8-5 说明这样定义的 $F(x,Y(x);h)$ 可作为区间 $[x,x+h]$ 上 Y' 的一个平均值.

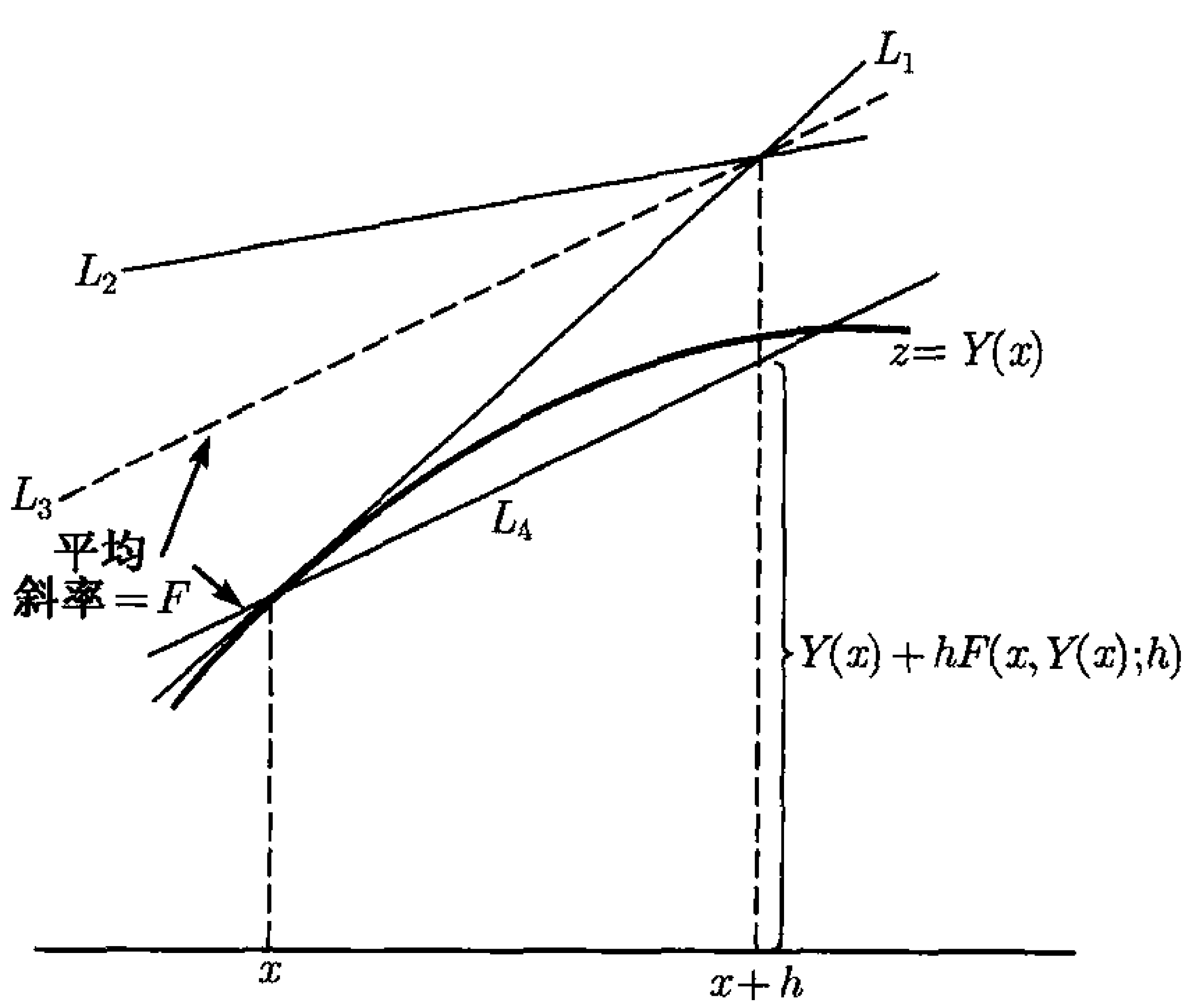


图 8-5 龙格–库塔方法 (8.81) 的一个说明; L_1 的斜率是 $f(x,Y(x))$, L_2 的斜率是 $f(x+h,Y(x)+hf(x,Y(x)))$, L_3 和 L_4 的斜率是平均 $F(x,Y(x);h)$

例 8.5.2　再次考虑问题 (8.66):

$$Y'(x) = -Y(x) + 2\cos(x), \quad Y(0) = 1$$

这里

$$f(x,y) = -y + 2\cos(x)$$

表 8-10 给出了利用 (8.81) 的数值结果. 龙格–库塔解的误差与用泰勒方法 (8.69) 得到的结果精度相当. 此外, 表 8-10 中的误差当 h 折半时大约减少到 1/4, 与方法的 2 阶收敛性相等. ■

表 8-10　二阶龙格–库塔方法的例

h	x	$y_h(x)$	误差
0.1	2.0	0.491 215 673	1.93E−3
	4.0	−1.407 898 629	−2.55E−3
	6.0	0.680 696 723	5.81E−5
	8.0	0.841 376 339	2.48E−3
	10.0	−1.380 966 579	−2.13E−3
0.05	2.0	0.492 682 499	4.68E−4
	4.0	−1.409 821 234	−6.25E−4
	6.0	0.680 734 664	2.01E−5
	8.0	0.843 254 396	6.04E−4
	10.0	−1.382 569 379	−5.23E−4

还可以得到更高阶的龙格–库塔方法. 一个流行的经典方法是下列四阶过程:

$$\begin{aligned} v_1 &= f(x_n, y_n) \\ v_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}v_1\right) \\ v_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}v_2\right) \\ v_4 &= f(x_n + h, y_n + hv_3) \\ y_{n+1} &= y_n + \frac{h}{6}[v_1 + 2v_2 + 2v_3 + v_4] \end{aligned} \tag{8.82}$$

这个方法的截断误差是 $O(h^5)$. 当用于解简单的微分方程

$$Y'(x) = f(x)$$

时 (其中 f 与 Y 无关), 这个方法化为 $[x_n, x_{n+1}]$ 上数值积分的辛普森 (Simpson) 法则. 在计算机或计算器上, 方法 (8.82) 容易编程, 而且数值结果比较精确. 这个方法的数值试验作为习题留给读者.

若初值问题的解 $Y(x)$ 和函数 $f(x,z)$ 充分可微, 并且龙格–库塔方法的截断误差是 $O(h^{p+1})$, 则可以证明龙格–库塔解满足

$$\max_{x_0\leqslant x_n\leqslant b}|Y(x_n)-y_h(x_n)|\leqslant ch^p \tag{8.83}$$

其中常数 c 与 $Y(x)$ 的导数和 $f(x,z)$ 的偏导数有关. 此外, 存在渐近误差公式, 其形式与泰勒方法 (8.75) 完全相同. 因而对龙格–库塔方法能使用理查森外推并能估计误差. 对二阶方法 (8.81), 我们有误差估计

$$Y(x_n)-y_h(x_n)\approx\frac{1}{3}\left[y_h(x_n)-y_{2h}(x_n)\right]$$

这同前面对二阶泰勒方法得到的结果一样, 见习题 3.

例 8.5.3 表 8-10 中对 $h=0.05$ 和 $x=10$ 估计误差. 我们有

$$Y(10)-y_{0.05}(10)\doteq\frac{1}{3}[-1.382\,566\,937\,9-(-1.380\,966\,579)]\doteq-5.34\text{E}-4$$

这个值与实际的误差 $-5.23\text{E}-4$ 很接近. ■

8.5.2 误差预报和控制

预报数值解 $y_h(x)$ 中的误差, 最容易的方法是使用理查森外推. 在给定的区间 $[x_0,b]$ 上, 分别用步长 $2h$ 和 h 求解初值问题. 然后使用理查森外推来估计误差. 如对二阶方法, 用了 $y_h(x)-y_{2h}(x)$ 来估计 $Y(x)-y_h(x)$, 见 (8.76). 这个方法估计误差的工作量与只计算 $y_h(x)$ 的工作量相比, 大约增加 50%. 这个工作量似乎很大, 但是除了太费时的问题外, 一般地它是值得付出的.

我们希望有一个在给定的区间 $[x_0,b]$ 上求解微分方程的计算机程序, 其误差小于给定的误差容限 $\varepsilon>0$. 遗憾的是, 对初值问题的大部分数值方法来说, 这是不可能的. 若在某些点 $\tilde{x}$ 上, 发现 $Y(\tilde{x})-y_h(\tilde{x})$ 太大, 则仅仅在 $\tilde{x}$ 往后的计算中减少 h 不可能使误差更小. 误差 $Y(\tilde{x})-y_h(\tilde{x})$ 依赖于 $\tilde{x}$ 左边所有节点上的误差累计. 因而, 为了减少 $\tilde{x}$ 处的误差, 必须以一个较小的步长 h 从 x_0 开始重复求解方程. 因为这个原因, 求解初值问题的大部分软件包程序不会直接控制误差, 虽然它们试图监视或界定误差. 而是使用间接的方法去影响误差的大小.

误差 $Y(x_n)-y_h(x_n)$ 称为 x_n 处的*整体误差*. 与之对比, x_n 处的截断误差 [见 (8.72)] 称为*局部误差*, 因为它是 x_n 步引入到解中的误差. 大部分含有误差控制的计算机程序基于估计局部误差, 然后通过适当地变化 h 来控制它, 这样做是希望保持整体误差充分小. 若给定误差参数 $\varepsilon>0$, 较好的程序选择步长 h, 保证局部误差 T_{n+1} 充分小, 通常满足

$$|T_{n+1}|\leqslant\varepsilon(x_{n+1}-x_n) \tag{8.84}$$

这样整体误差也小, 对许多微分方程, 整体误差小于 $\varepsilon(x_{n+1}-x_0)$.

我们可以使用各种各样的技巧, 包括理查森外推法, 来估计局部误差. 最近设计出的另一种方法, 导致当前最流行的龙格–库塔方法. 我们利用两个不同阶的方法同时计算, 而不是计算一个固定阶的方法. 然后, 用较高阶公式去估计较低阶公式中的误差. 这些方法称为**菲尔伯格** (Fehlberg) **方法**, 我们将给出四阶和五阶公式组成的这样的方法.

在 $[x_n, x_{n+1}]$ 内定义 6 个中间斜率

$$\begin{aligned} v_0 &= f(x_n, y_n) \\ v_i &= f\left(x_n + a_i h, y_n + h\sum_{j=0}^{i-1} b_{ij} v_j\right), \quad i = 1,2,3,4,5 \end{aligned} \tag{8.85}$$

然后由下式给出四阶公式和五阶公式

$$y_{n+1} = y_n + h\sum_{i=0}^{4} c_i v_i \tag{8.86}$$

$$\hat{y}_{n+1} = y_n + h\sum_{i=0}^{5} d_i v_i \tag{8.87}$$

表 8-11 和表 8-12 给出系数 a_i, b_{ij}, c_i, d_i.

表 8-11 菲尔伯格系数 a_i, b_{ij}

i	a_i	b_{i0}	b_{i1}	b_{i2}	b_{i3}	b_{i4}
1	1/4	1/4				
2	3/8	3/32	9/32			
3	12/13	1 932/219 7	−7 200/2 197	7 296/2 197		
4	1	439/216	−8	3 680/513	−845/4 104	
5	1/2	−8/27	2	−3 544/2 565	1 859/4 104	−11/40

表 8-12 菲尔伯格系数 c_i, d_i

i	0	1	2	3	4	5
c_i	25/216	0	1 408/2 565	2 197/4 104	−1/5	
d_i	16/135	0	6 656/12 825	28 561/56 430	−9/50	2/55

四阶公式 (8.86) 的局部误差由下式估计

$$T_{n+1} \approx \hat{y}_{n+1} - y_{n+1} \tag{8.88}$$

可以证明当 $h \to 0$ 时, 这是一个正确的渐近结果. 利用这个估计, 若 T_{n+1} 太小或太大, 可以变化步长以得到合适的 T_{n+1} 值. 注意到公式 (8.86) 和 (8.87) 使用公共的中间斜率 $v_0, \cdots, v_4$. 这样, 每步只需要计算 6 个中间斜率.

例 8.5.4 解

$$Y'(x) = -Y(x) + 2\cos(x), \quad Y(0) = 1 \tag{8.89}$$

其真解是 $Y(x) = \sin(x) + \cos(x)$. 表 8-13 给出 $h = 0.25$ 和 $h = 0.125$ 时的数值结果. 把整体误差与表 8-9 和表 8-10 中使用的二阶方法的误差作比较. 可以看出 y_h 中的整体误差减少到原来的 $\dfrac{1}{21} \sim \dfrac{1}{17}$, 这相当接近于四阶方法的理论值 1/16. 由 (8.88) 估计的截断误差也列在表中以显示它们与整体误差的确不同. ■

表 8-13 四阶菲尔伯格公式 (8.86) 的例

h	x	$y_h(x)$	$Y(x) - y_h(x)$	$\hat{y}_h(x) - y_h(x)$
0.25	2.0	0.493 156 301	−5.71E−6	−9.49E−7
	4.0	−1.410 449 823	3.71E−6	1.62E−6
	6.0	0.680 752 304	2.48E−6	−3.97E−7
	8.0	0.843 864 007	−5.79E−6	−1.29E−6
	10.0	−1.383 094 975	2.34E−6	1.47E−6
0.125	2.0	0.493 150 889	−2.99E−7	−2.35E−8
	4.0	−1.410 446 334	2.17E−7	4.94E−8
	6.0	0.680 754 675	1.14E−7	−1.76E−8
	8.0	0.843 858 525	−3.12E−7	−3.47E−8
	10.0	−1.383 092 786	1.46E−7	4.65E−8

方法 (8.85)~(8.88) 使用 $\hat{y}_{n+1}$ 仅仅是为了估计四阶方法中的截断误差. 实际上, 可保留 $\hat{y}_{n+1}$ 而不是 y_{n+1} 作为数值解, 因而在 (8.85)~(8.87) 的右边, 应该用 $\hat{y}_n$ 代替 y_n. (8.88) 中的量仍为四阶方法的截断误差. 基于这点的程序将是五阶的, 但程序应改变它们的步长以控制四阶方法中的局部误差. 这样做有助于使这些程序的整体误差非常准确.

例 8.5.5 重复上述例子, 但使用前面描述的五阶方法. 结果在表 8-14 中给出. 当 h 折半时误差大约减少到原来的 1/32, 这与五阶方法一致. ■

表 8-14 五阶方法 (8.87) 的例

h	x	$\hat{y}_n(x)$	$Y(x) - \hat{y}_n(x)$
0.25	2.0	0.493 151 148	−5.58E−7
	4.0	−1.410 446 359	2.43E−7
	6.0	0.680 754 463	3.26E−7
	8.0	0.843 858 731	−5.18E−7
	10.0	−1.383 092 745	1.05E−7
0.125	2.0	0.493 150 606	−1.61E−8
	4.0	−1.410 446 124	8.03E−9
	6.0	0.680 754 780	8.65E−9
	8.0	0.843 858 228	−1.53E−8
	10.0	−1.383 092 644	4.09E−9

8.5.3 MATLAB 内置函数

MATLAB 内置函数ode45是一个自适应算法, 它类似于基于 (4, 5) 对的菲尔伯格方法 (8.86) 和 (8.87). 它是多门德–普林斯 (Dormand-Prince)(4, 5) 对 [14]. 与本章中至今的讨论不同, ode45适当地调整步长使得估计的局部误差, 其绝对误差在给定的界AbsTol之中, 而相对误差在另一个界RelTol之中. 误差容限是

$$\texttt{AbsTol} = 10^{-6}, \quad \texttt{RelTol} = 10^{-3}$$

对初值问题 (8.7), 可以用

```
[x,y] = ode45('fcn',[x0,b],Y0)
```

调用ode45函数, 其中fcn是计算微分方程右端函数 $f(x, z)$ 的 m 文件名. 若x0 = 0, 则可简写成

```
[x,y] = ode45('fcn',b,Y0)
```

例 8.5.6 再次考虑问题 (8.89) 并使用 MATLAB 内置函数ode45求解. 用

```
z = -y + 2 * cos(x);
```

定义右端函数z = fcn(x,y); 并利用

```
[x,y] = ode45('fcn',[0,10],1);
plot(x,y,'*-')
```

生成数值解的图形. 图形在图 8-6 中给出. 为了得到 $x = 2, 4, 6, 8, 10$ 处的解值, 定义

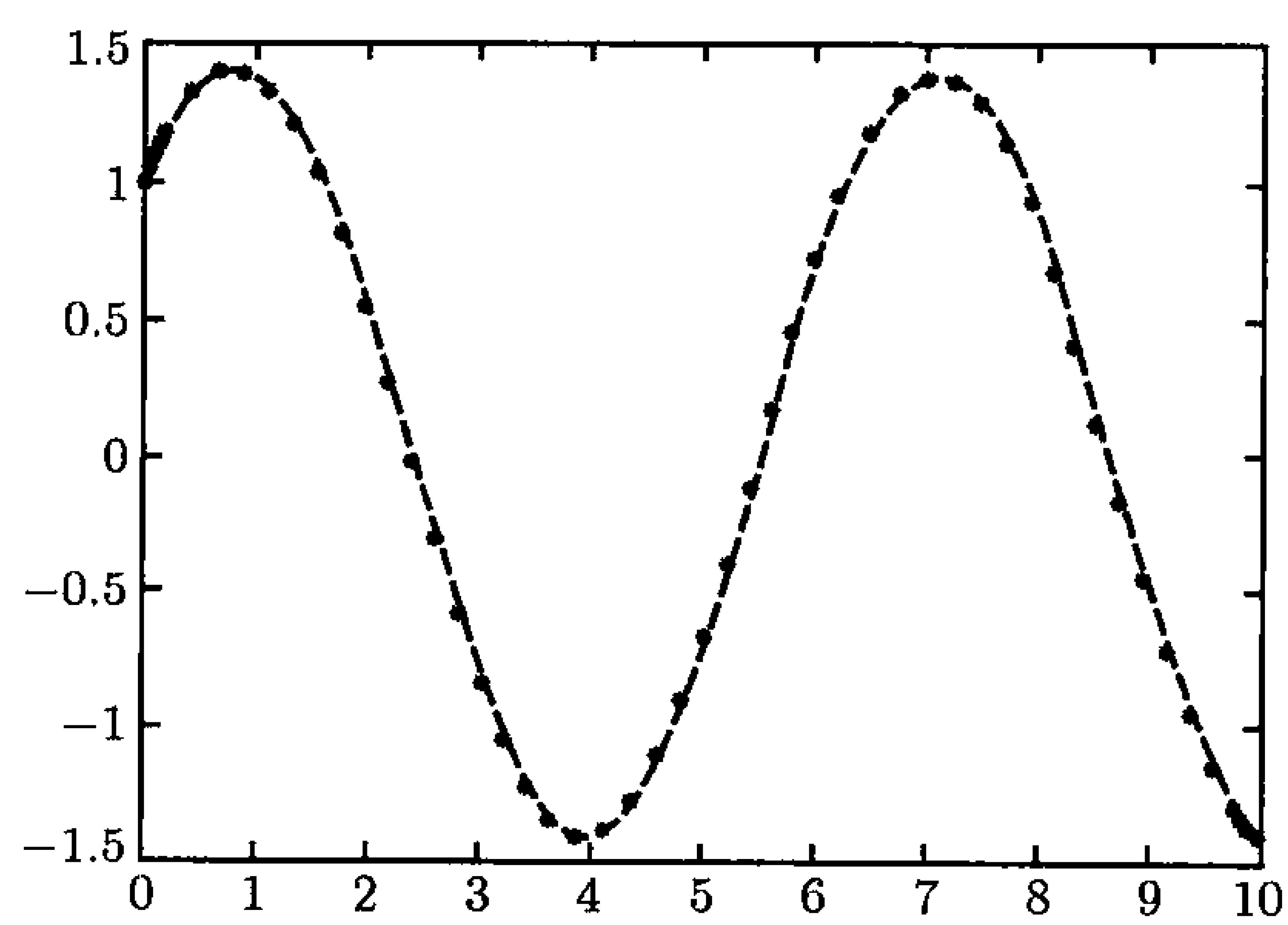


图 8-6 用ode45的 (8.89) 的数值解

```
xvalues = 0:2:10;
```

语句

$$[\mathtt{x},\mathtt{y}] = \mathtt{Ode45}(\text{'fcn'},\mathtt{xvaule},1);$$

$$\mathtt{err} = \mathtt{sin}(\mathtt{x}) + \mathtt{cos}(\mathtt{x}) - \mathtt{y};$$

将在指定的点 $x = 0,2,4,6,8,10$ 处, 计算数值解的值及它们的误差. 表 8-15 中给出数值结果.

表 8-15　带误差容限的ode45例子

x	$y_h(x)$	$Y(x) - y_h(x)$
2.0	0.492 869 823	2.81E−4
4.0	−1.411 610 153	1.16E−3
6.0	0.680 895 458	−1.41E−4
8.0	0.843 670 835	1.87E−4 .
10.0	−1.383 138 747	4.61E−5

如果要求更高的解精度, 我们可以在使用数据结构options和odeset函数的地方指定较小的误差容限. 例如, 定义

```
options = odeset('AbsTol', 1e - 9, 'Reltol', 1e - 7);
```

然后使用

```
[x,y]=ode 45 ('fcn',xvaules,1,options);
err = sin(x) + cos(x) - y;
```

在指定的点 $x = 0,2,4,6,8,10$ 处, 计算数值解值和它们的误差. 表 8-16 中出给数值结果. ■

表 8-16　带较小误差容限的ode45例子

x	$y_h(x)$	$Y(x) - y_h(x)$
2.0	0.493 150 575	1.56E−8
4.0	−1.410 446 139	2.28E−8
6.0	0.680 754 810	−2.17E−8
8.0	0.843 858 213	−2.8E−10
10.0	−1.383 092 630	−1.01E−8

另一个 MATLAB 内置函数ode23使用 [8] 中引进的博加斯基–沙姆皮尼 (Bogacki-Shampine)(2, 3) 对. 对非刚性常微分方程, 程序ode45和ode23是有效的. 对刚性方程, 可以使用实施隐式龙格–库塔算法的 MATLAB 内置函数ode23tb. ode23函数和ode23tb函数的使用完全地类似于ode45, 鼓励读者对它们进行试验.

习题

1. 利用推导 (8.69) 的相同过程可以得到问题 (8.66) 的三阶泰勒方法. 以三阶泰勒逼近

$$Y(x_{n+1}) \approx Y(x_n) + hY'(x_n) + \frac{h^2}{2}Y''(x_n) + \frac{h^3}{6}Y'''(x_n)$$

为基础, 推导数值方法

$$\begin{aligned} y_{n+1} &= y_n + h[-y_n + 2\cos(x_n)] + \frac{h^2}{2}[y_n - 2\cos(x_n) - 2\sin(x_n)] \\ &\quad + \frac{h^3}{6}[-y_n + 2\sin(x_n)], \quad n \geqslant 0 \end{aligned} \tag{8.90}$$

用方法 (8.90) 去求解问题 (8.66). 对 $0 \leqslant x \leqslant 10$, 用步长 $h = 0.1, 0.05$ 计算. 与表 8-9 中的值作比较, 检验 h 折半时误差减少的比率.

提示: 为使程序设计更容易, 可修改 8.2 节中给出的欧拉程序.

2. 用二阶泰勒方法计算下列问题的解. 使用步长 $h = 0.2, 0.1, 0.05$.

(a) $Y'(x) = [\cos(Y(x))]^2, \quad 0 \leqslant x \leqslant 10, \quad Y(0) = 0;$

$$Y(x) = \tan^{-1}(x)$$

(b) $Y'(x) = \dfrac{1}{1+x^2} - 2[Y(x)]^2, \quad 0 \leqslant x \leqslant 10, \quad Y(0) = 0;$

$$Y(x) = \frac{x}{1+x^2}$$

(c) $Y'(x) = \dfrac{1}{4}Y(x)\left[1 - \dfrac{1}{20}Y(x)\right], \quad 0 \leqslant x \leqslant 20, \quad Y(0) = 1;$

$$Y(x) = \frac{20}{1 + 19\mathrm{e}^{-x/4}}$$

(d) $Y'(x) = -[Y(x)]^2, \quad 1 \leqslant x \leqslant 10, \quad Y(1) = 1; \quad Y(x) = \dfrac{1}{x}$

这些问题已经在 8.2 节习题 1 求解过. 把现在的结果与前面那些结果作比较.

3. 回顾 (8.75) 给出的泰勒方法的渐近误差. 对二阶方法,

$$Y(x_n) - y_h(x_n) = h^2 D(x_n) + O(h^3)$$

由此推导理查森外推公式

$$\begin{aligned} Y(x_n) &= \frac{1}{3}[4y_h(x_n) - y_{2h}(x_n)] + O(h^3) \\ &\approx \frac{1}{3}[4y_h(x_n) - y_{2h}(x_n)] \equiv \tilde{y}_h(x_n) \end{aligned}$$

和渐近误差估计

$$\begin{aligned} Y(x_n) - y_h(x_n) &= \frac{1}{3}[y_h(x_n) - y_{2h}(x_n)] + O(h^3) \\ &\approx \frac{1}{3}[y_h(x_n) - y_{2h}(x_n)] \end{aligned}$$

提示: 考虑公式

$$Y(x_n) - y_{2h}(x_n) = 4h^2 D(x_n) + O(h^3)$$

并把它与前面 $Y(x_n) - y_h(x_n)$ 的公式适当地组合.

4. 对阶 $p \geqslant 1$ 的方法重复习题 3. 推导公式

$$Y(x_n) \approx \frac{1}{2^p - 1}\left[2^p y_h(x_n) - y_{2h}(x_n)\right] \equiv \tilde{y}_h(x_n)$$

证明 $\tilde{y}_h(x_n)$ 的误差阶为 $p+1$, 并且

$$Y(x_n) - y_h(x_n) \approx \frac{1}{2^p - 1}\left[y_h(x_n) - y_{2h}(x_n)\right]$$

5. 对 $h = 0.05$, 利用习题 3 估计表 8-9 中结果的误差. 再产生理查森外推 $\tilde{y}_h(x_n)$ 并计算它的误差. 把它的精度与 $y_h(x_n)$ 的精度作比较.
6. 对应于 (8.78) 中 $\gamma_2 = \frac{3}{4}$ 和 $\gamma_2 = 1$ 推导 2 阶龙格–库塔方法 (8.77). 类似于图 8-5 关于 $\gamma_2 = \frac{1}{2}$ 的情况, 对 $\gamma_2 = 1$, 画出有助于说明的图形.
7. 用习题 6 的一个公式求解问题 (8.66). 把你的结果与表 8-10 关于 $\gamma_2 = \frac{1}{2}$ 的公式 (8.81) 的那些结果作比较.
8. 利用 (8.81) 求解习题 2 中的方程. 利用习题 3 的结果估计误差, 并把它与真实的误差作比较.
9. 实施经典过程 (8.82), 并把它应用于方程 (8.66). 用步长 $h = 0.25$ 和 $h = 0.125$ 求解. 把结果与表 8-13 中 4 阶菲尔伯格例子的结果作比较.
   提示: 可修改 8.2 节的欧拉程序
10. 利用习题 9 的程序求解习题 2 的方程.
11. 用内置函数**ode45**求解习题 2 的方程. 选择几个误差容限作实验.
12. 用内置函数**ode23**求解习题 2 的方程. 选择几个误差容限作实验.
13. 考虑在地球引力场中质量为 m 的质点垂直下落的运动, 并假设由摩擦力 $p(v)$ 对抗的向下运动依赖于质点的速度 $v(t)$. 则速度满足方程

$$mv'(t) = -mg + p(v), \quad t \geqslant 0, \quad v(0)\text{已知}$$

   设 $m = 1$ 千克, $g = 9.8\text{米/秒}^2, v(0) = 0$. 对 $0 \leqslant t \leqslant 20$ 以及下列选取的 $p(v)$ 解方程:
   (a) $p(v) = -0.1v$, 它对落体是正的.
   (b) $p(v) = 0.1v^2$.
   求至少有三位精确数字的答案. 画出函数 $v(t)$ 的图形. 比较这些解.
14. (a) 利用龙格–库塔方法 (8.81), 解

$$Y'(x) = -Y(x) + x^{0.1}\left[1.1 + x\right], \quad Y(0) = 0$$

   真解是 $Y(x) = x^{1.1}$. 在 $[0, 5]$ 上解方程, 打印 $x = 1, 2, 3, 4, 5$ 处的解及误差. 使用步长 $h = 0.1, 0.05, 0.025, 0.0125, 0.00625$. 计算 h 折半时误差减少的比率. 把这个比率与理论上的收敛率 $O(h^2)$ 比较, 情况如何? 尽可能好地说明你的结果.
   (b) 用阶数 $\geqslant 2$ 的泰勒方法求解 (a) 中的方程会有什么困难? 它告诉我们有关解的什么情况?

8.6　多　步　法

泰勒法和龙格–库塔方法都是单步法, 因为 y_{n+1} 由 y_n 单独确定. 本节考虑使用前面几个节点处解的值来计算数值解 y_{n+1} 的多步法. 这里叙述两族多步法的推导.

通过在区间 $[x_n, x_{n+1}]$ 上对微分方程

$$Y'(x) = f(x, Y(x))$$

积分, 得到

$$\int_{x_n}^{x_{n+1}} Y'(x)\mathrm{d}x = \int_{x_n}^{x_{n+1}} f(x, Y(x))\mathrm{d}x$$

$$Y(x_{n+1}) = Y(x_n) + \int_{x_n}^{x_{n+1}} f(x, Y(x))\mathrm{d}x \tag{8.91}$$

通过对 (8.91) 近似积分就可以推导求解 $Y(x)$ 的数值方法. 有许多这样的方法, 但我们只考虑最流行的亚当斯–巴什福思 (Adams-Bashforth)(AB) 方法和亚当斯–莫尔顿 (Adams-Moulton)(AM) 方法. 这些方法是求解初值问题中最广泛使用的计算机代码的基础. 特别地, 当我们希望求高精度的解或导数 $f(x, z)$ 值计算的代价昂贵时, 它们一般地比龙格–库塔方法更有效.

为计算积分

$$\int_{x_n}^{x_{n+1}} g(x)\mathrm{d}x, \quad g(x) = f(x, Y(x)) \tag{8.92}$$

的值, 我们利用多项式插值逼近 $g(x)$, 然后积分插值多项式. 对给定的非负整数 q, AB 方法用点集 $\{x_n, x_{n-1}, \cdots, x_{n-q}\}$ 上的 q 次插值多项式, 而 AM 方法用点集 $\{x_{n+1}, x_n, x_{n-1}, \cdots, x_{n-q+1}\}$ 上的 q 次插值多项式. 首先考虑 AB 方法, 从基于线性插值的 AB 方法入手.

在 $\{x_n, x_{n-1}\}$ 处的线性插值 $g(x)$ 的多项式是

$$p_1(x) = \frac{1}{h}[(x_n - x)g(x_{n-1}) + (x - x_{n-1})g(x_n)] \tag{8.93}$$

在 $[x_n, x_{n+1}]$ 上积分, 得到

$$\int_{x_n}^{x_{n+1}} g(x)\mathrm{d}x \approx \int_{x_n}^{x_{n+1}} p_1(x)\mathrm{d}x = \frac{3h}{2}g(x_n) - \frac{h}{2}g(x_{n-1})$$

利用类似于 5.1 节和 5.2 节使用过的那些想法, 可以得到更完整的结果

$$\int_{x_n}^{x_{n+1}} g(x)\mathrm{d}x = \frac{h}{2}[3g(x_n) - g(x_{n-1})] + \frac{5}{12}h^3 g''(\xi_n) \tag{8.94}$$

其中 ξ_n 满足 $x_{n-1} \leqslant \xi_n \leqslant x_{n+1}$, 也可见习题 3. 把这个式子应用于 (8.91) 中, 给出

$$Y(x_{n+1}) = Y(x_n) + \frac{h}{2}[3f(x_n, Y(x_n)) - f(x_{n-1}, Y(x_{n-1}))] + \frac{5}{12}h^3 Y'''(\xi_n)$$

去掉最后的截断误差项, 得到数值方法

$$y_{n+1} = y_n + \frac{h}{2}[3f(x_n, y_n) - f(x_{n-1}, y_{n-1})] \tag{8.95}$$

注意到只能对 $n \geqslant 1$ 应用这个方法. 求 y_2 时需要 y_0 和 y_1, 而 y_1 不能从 (8.95) 中求出. 所以必须用另外的方法得到 y_1 的值. 方法 (8.95) 是二步法的一个例子, 因为求 x_{n+1} 处的值时需要 x_{n-1} 和 x_n 处的值. 若可以确定精度为 $O(h^2)$ 的 y_1(例如, 通过欧拉方法), 则 AB 方法 (8.95) 的阶数为 2, 即它的整体误差以 $O(h^2)$ 为界, 因为截断误差以 $O(h^3)$ 为界.

例 8.6.1 利用 (8.95) 求解

$$Y'(x) = -Y(x) + 2\cos(x), \quad Y(0) = 1 \tag{8.96}$$

真解为 $Y(x) = \sin(x) + \cos(x)$. 我们取 $y_1 = Y(x_1)$, 这仅仅是为了有助于说明. 表 8-17 给出 $h = 0.05$ 时的数值结果. 注意当 h 折半时, 误差大约减少到原来的 1/4, 这与二阶的数值方法相容. 对二阶方法利用 (8.76) 中的估计, 表中也包含理查森误差估计. 在误差以速度 $O(h^2)$ 减少之处, 误差估计较为准确. ■

表 18-17 二阶亚当斯–巴什福思方法的例

x	$y_n(x)$	$Y(x) - y_{2h}(x)$	$Y(x) - y_h(x)$	比率	$\frac{1}{3}[y_h(x) - y_{2h}(x)]$
2	0.492 597 22	2.13E−3	5.53E−4	3.9	5.26E−4
4	−1.411 169 63	2.98E−3	7.24E−4	4.1	7.52E−4
6	0.681 742 79	−3.91E−3	−9.88E−4	4.0	−9.73E−4
8	0.843 736 78	3.68E−4	1.21E−4	3.0	8.21E−5
10	−1.383 982 54	3.61E−3	8.90E−4	4.1	9.08E−4

通常认为 AB 方法的工作量比龙格–库塔方法少, 把 (8.95) 与 (8.81) 的二阶龙格–库塔方法作比较就可以找到主要原因. 用二阶龙格–库塔方法, 从 x_n 到 x_{n+1} 每步求两个 f 的值. 与之对比, 倘若重新使用 f 以前的值, 则 AB 公式 (8.95) 每步只需求一个值. 有一些其他的因素影响数值方法的选择, 但是, 对给定的精度需要多次求 f 的值时, AB 方法和 AM 方法一般更为有效.

伴随多步法出现的问题是必须用另外的方法生成解的若干个初始值. 对 (8.95) 的二阶 AB 方法, 必须得到 y_1. 因为 $y_h(x_n)$ 的整体误差是 $O(h^2)$, 所以必须保证 $Y(x_1) - y_h(x_1)$ 也是 $O(h^2)$. 利用前几节的方法, 有两种直接的可能性.

情况 (i) 利用欧拉方法:

$$y_1 = y_0 + hf(x_0, y_0) \tag{8.97}$$

假设 $y_0 = Y_0$, 根据 (8.27), 我们确定这个方法的误差为

$$Y(x_1) - y_1 = \frac{h^2}{2} Y''(\xi_1)$$

因而, (8.97) 符合选取 y_1 的误差标准. 总体上, 欧拉方法只有 $O(h)$ 的精度, 但是单步的误差是 $O(h^2)$.

情况 (ii) 利用例如 (8.81) 那样的二阶龙格–库塔方法. 因为只使用了 x 的一步, 所以 $Y(x_1) - y_1$ 为 $O(h^3)$, 这个误差足够小.

例 8.6.2 利用 (8.97) 与 (8.95) 求解问题 (8.96). 对 $h = 0.05$ 和 $x = 10$, 数值解的误差是

$$Y(10) - y_h(10) \doteq 8.90\text{E}-4$$

这和表 8-17 中的结果几乎相同. ■

在 (8.92) 的近似积分中使用高次多项式插值法就可以得到高阶亚当斯–巴什福思方法. 例如, 设 $p_2(x)$ 表示 x_n, x_{n-1}, x_{n-2} 处插值 $g(x)$ 的二次多项式. 然后使用

$$\int_{x_n}^{x_{n+1}} g(x)\mathrm{d}x \approx \int_{x_n}^{x_{n+1}} p_2(x)\mathrm{d}x$$

积分此式并且包含 (用其他手段得到) 的误差项, 得到

$$\int_{x_n}^{x_{n+1}} g(x)\mathrm{d}x = \frac{h}{12}\left[23g(x_n) - 16g(x_{n-1}) + 5g(x_{n-2})\right] + \frac{3}{8}h^4 g'''(\xi_n)$$

其中 ξ_n 满足 $x_{n-2} \leqslant \xi_n \leqslant x_{n+1}$. 把此式用于微分方程的积分表达式 (8.91), 得到

$$\begin{aligned} Y(x_{n+1}) =& Y(x_n) + \frac{h}{12}[23f(x_n, Y(x_n)) - 16f(x_{n-1}, Y(x_{n-1})) \\ &+ 5f(x_{n-2}, Y(x_{n-2}))] + \frac{3}{8}h^4 Y^{(4)}(\xi_n) \end{aligned}$$

舍弃最后的截断误差项, 得到三阶 AB 方法

$$y_{n+1} = y_n + \frac{h}{12}[23y'_n - 16y'_{n-1} + 5y'_{n-2}], \quad n \geqslant 2 \tag{8.98}$$

其中 $y'_k \equiv f(x_k, y_k)$, $k \geqslant 0$. 这是三步法, 要求 $n \geqslant 2$. 初始值 y_1, y_2 必须分别用其他方法得到.

一般地, 可以证明基于 q 次插值的 AB 方法是 $q+1$ 步方法, 并且它的截断误差具有形式

$$T_{n+1} = c_q h^{q+2} Y^{(q+2)}(\xi_n)$$

其中 ξ_n 满足 $x_{n-q} \leqslant \xi_n \leqslant x_{n+1}$. 初值 $y_1, \cdots, y_q$ 必须用其他方法生成. 若这些初值中的误差满足

$$Y(x_n) - y_n(x_n) = O(h^{q+1}), \quad n = 1, 2, \cdots, q \tag{8.99}$$

则 $q+1$ 步 AB 方法的整体误差也是 $O(h^{q+1})$. 此外, 整体误差也具有类似于 8.5 节中所讨论过的泰勒方法和龙格–库塔方法的渐近误差公式

$$Y(x_n) - y_h(x_n) = D(x_n)h^{q+1} + O(h^{q+2})$$

因而可用理查森外推加速方法的收敛性和估计误差.

为生成 $(q+1)$ 步 AB 方法的初值 $y_1, \cdots, y_q$, 并使它们的误差满足条件 (8.99), 使用 q 阶龙格–库塔方法就足够了. 然而, 在许多情况下, 人们更喜欢使用更精确的 $q+1$ 阶龙格–库塔方法, 它与 $(q+1)$ 步 AB 方法有相同的阶. 亚当斯方法有其他一些计算机程序所用的过程, 但不在此讨论它们.

表 8-18 中给出一阶到四阶的 AB 方法. 一阶公式就是欧拉方法. 在表中, $y'_k \equiv f(x_k, y_k)$.

表 8-18 亚当斯–巴什福思方法

q	阶	方 法	截断误差
0	1	$y_{n+1} = y_n + hy'_n$	$\frac{1}{2}h^2Y''(\xi_n)$
1	2	$y_{n+1} = y_n + \frac{h}{2}[3y'_n - y'_{n-1}]$	$\frac{5}{12}h^3Y'''(\xi_n)$
2	3	$y_{n+1} = y_n + \frac{h}{12}[23y'_n - 16y'_{n-1} + 5y'_{n-2}]$	$\frac{3}{8}h^4Y^{(4)}(\xi_n)$
3	4	$y_{n+1} = y_n + \frac{h}{24}[55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3}]$	$\frac{251}{720}h^5Y^{(5)}(\xi_n)$

例 8.6.3 利用四阶 AB 方法解问题 (8.96). 利用真解生成初始值 y_1, y_2, y_3,

$$y_i = Y(x_i), \quad i = 1, 2, 3$$

表 8-19 给出 $h = 0.125$ 和 $2h = 0.25$ 的计算结果. 最后一列中给出四阶方法的理查森误差估计. 对四阶方法, 当 h 折半时误差大约减少 16 倍. 这种情况下, 理查森误差估计是准确的. 没有出现理查森误差估计严重低估误差的情况. ■

表 8-19 四阶亚当斯–巴什福思方法的例子

x	$y_h(x)$	$Y(x) - y_{2h}(x)$	$Y(x) - y_h(x)$	比率	$\frac{1}{15}[y_h(x) - y_{2h}(x)]$
2	0.493 186 80	−3.96E−4	−3.62E−5	10.9	−2.25E−5
4	−1.410 376 98	−1.25E−3	−6.91E−5	18.1	−7.37E−5
6	0.680 679 62	1.05E−3	7.52E−5	14.0	6.12E−5
8	0.843 854 16	3.26E−4	4.06E−6	80.0	2.01E−5
10	−1.383 013 76	−1.33E−3	−7.89E−5	16.9	−7.82E−5

把这些结果与表 8-13 中四阶菲尔伯格方法的结果作比较, 可以看出四阶 AB 方法的误差似乎大很多. 但是注意到菲尔伯格公式对 x_n 到 x_{n+1} 的每一步需求 5 个 $f(x, z)$ 的值; 假定利用以前的值, 则四阶 AB 方法每步只需求一个 f 的值. 如果这个 AB 方法使用 $(1/5)h$ 作步长 (为了有类似的 f 求值次数), 则数值解的误差

将大约减少到原来的 $\frac{1}{5^4}=\frac{1}{625}$. 因而在工作量相当的条件下 (以 f 求值次数度量), AB 方法的误差通常小于表 8-13 中菲尔伯格方法的误差.

如同 AB 方法那样, 我们从基于线性插值的方法入手介绍 AM 方法. 设 $p_1(x)$ 是在 x_n 和 x_{n+1} 处插值 $g(x)$ 的线性多项式

$$p_1(x)=\frac{1}{h}\left[(x_{n+1}-x)g(x_n)+(x-x_n)g(x_{n+1})\right]$$

利用它作 (8.92) 的近似积分, 有

$$\int_{x_n}^{x_{n+1}} g(x)\mathrm{d}x \approx \int_{x_n}^{x_{n+1}} p_1(x)\mathrm{d}x=\frac{h}{2}\left[g(x_n)+g(x_{n+1})\right]$$

这正好是 5.1 节介绍的梯形法. 把这个式子用于积分公式 (8.91), 并且包括梯形法的误差项 (5.26), 得到

$$Y(x_{n+1})=Y(x_n)+\frac{h}{2}[f(x_n,Y(x_n))+f(x_{n+1},Y(x_{n+1}))]-\frac{h^3}{12}Y'''(\xi_n) \tag{8.100}$$

舍弃最后的截断误差项, 得到 AM 方法

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1})],\quad n\geqslant 0 \tag{8.101}$$

这正是 8.4 节研究过的梯形法. 它是一个二阶方法, 即它的整体误差是 $O(h^2)$.

例 8.6.4　利用 AM 方法 (8.101)(梯形法) 求解前面的例子 (8.96). 表 8-20 给出 $h=0.05$, $2h=0.1$ 时的数值结果. 此表的最后一列给出二阶方法的理查森误差估计. 在此情形, $O(h^2)$ 误差性质非常明显, 并且误差估计十分准确. ■

表 8-20　二阶亚当斯–莫尔顿方法的例子

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$
2	−4.59E−4	−1.15E−4	4.0	−1.15E−4
4	−5.61E−4	−1.40E−4	4.0	−1.40E−4
6	7.98E−4	2.00E−4	4.0	2.00E−4
8	−1.21E−4	−3.04E−5	4.0	−3.03E−4
10	−7.00E−4	−1.75E−4	4.0	−1.75E−4

例 8.6.5　重复上面的例子, 但用 8.4 节所述的过程, 对每个 n 只计算一次迭代. 则与表 8-20 给出的那些值比较, 误差没有大的改变. 例如, 对 $x=10$ 和 $h=0.05$, 误差是

$$Y(10)-y_h(10)\doteq -2.02\mathrm{E}-4$$

这个值与表 8-20 给出的值 $-1.75\mathrm{E}-4$ 差别不大. 使用迭代 $y_{n+1}^{(1)}$ 作为根 y_{n+1}, 对大部分微分方程的解不会有很大的影响. ■

通过对 (8.92) 的函数 $g(x)$ 在节点 $\{x_{n+1}, x_n, \cdots, x_{n-q+1}\}$ 处插值的 q 次多项式积分, 可得到 $q+1$ 阶 AM 方法. 它是一个隐式方法, 但其他有关的理论与前述的 AB 方法相同. 表 8-21 给出一阶到四阶的 AM 方法. 表中 $y'_k \equiv f(x_k, y_k)$. 注意, 一阶 AM 方法是向后欧拉方法, 而二阶 AM 方法是梯形方法.

表 8-21 亚当斯–莫尔顿方法

q	阶	方 法	截断误差
0	1	$y_{n+1} = y_n + hy'_{n+1}$	$-\frac{1}{2}h^2Y''(\xi_n)$
1	2	$y_{n+1} = y_n + \frac{h}{2}[y'_{n+1} + y'_n]$	$-\frac{1}{12}h^3Y'''(\xi_n)$
2	3	$y_{n+1} = y_n + \frac{h}{12}[5y'_{n+1} + 8y'_n - y'_{n-1}]$	$-\frac{1}{24}h^4Y^{(4)}(\xi_n)$
3	4	$y_{n+1} = y_n + \frac{h}{24}[9y'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2}]$	$-\frac{19}{720}h^5Y^{(5)}(\xi_n)$

假定利用以前计算过的函数值, 则 AM 方法的工作量是每步求两个 $f(x,z)$ 的值. 这里包含对预估求一个 f 的值, 然后对 (8.59) 那样的 AM 方法, 在迭代公式中求一个 f 的值. 如果这样做, 则在同等的工作量下, AM 方法并不会比同阶的 AB 方法好多少. 但是 AM 方法有一些其他的性质使它们令人满意地用于许多类型的微分方程. 令人满意的特征与数值方法的稳定性特性有关. 正如以前已经知道的, 一阶和二阶 AM 方法绝对稳定, 因而它们特别适合解刚性微分方程.

对具有刚才所讨论的数值稳定性而言, 绝对稳定性区域越大, 对 h 的限制条件越少. 因此, 倘若两个方法的精度相当的话, 一般地我们使用具有大的绝对稳定性区域的方法. 可以证明, 对于同阶的 AB 方法和 AM 方法, AM 方法具有较大的绝对稳定性区域. 因此, 亚当斯–莫尔顿方法一般比亚当斯–巴什福思方法更适合于应用.

例 8.6.6 二阶 AB 方法以 $-1 < h\lambda < 0$ 作为其稳定性区域的实部, 而二阶 AM 方法以 $-\infty < h\lambda < 0$ 作为其稳定性区域的实部. 这个 AM 方法对 h 没有稳定性限制. ■

如前所述, 求解初值问题的一些最流行的计算机程序是基于组合使用 AM 方法和 AB 方法的. 这些程序通过改变步长 h 和改变方法的阶来控制截断误差. 允许可能的阶如 12 一样大或更大些; 当解 $Y(x)$ 有几次连续导数并且 $Y(x)$ 缓慢地变化时, 这产生非常有效的数值方法. 关于亚当斯方法完整的讨论和计算机程序的例子在 Shampine(1994)[31] 中给出.

基于多步法的内置 MATLAB 程序是 `ode113` 和 `ode15s`. 这些程序分别实施不同阶数的显式和隐式线性多步法. 程序 `ode113` 用于求解非刚性常微分方程, 而 `ode15s` 用于求解刚性常微分方程. 这些程序的使用方式与前节讨论过的程序 `ode45` 用法相同.

MATLAB 程序 为有助于本节方法的程序设计, 我们提供 8.2 节欧拉程序的一个

修正. 程序实施 (8.95) 中给出的二阶亚当斯–巴什福思公式, 初值 y_1 按 (8.97) 的欧拉法产生.

```
function [x,y] = AB2(x0,y0,x_end,h,fcn)
%
% function [x,y]=AB2(x0,y0,x_end,h,fcn)
%
% Solve the initial value problem
%    y' = f(x,y),  x0 <= x <= b,  y(x0)=y0
% Use Adams-Bashforth formula of order 2 with
% a stepsize of h. Euler's method is used for
% the value y1.  The user must supply a program
% with some name, say, deriv, and a first  line of
% the form
%   function ans=deriv(x,y)
% A sample call would be
%   [t,z]=AB2(t0,z0,b,delta,'deriv')
%
% Output:
% The routine AB2 will return two vectors, x and y.
% The vector x will contain the node points
%   x(1)=x0, x(j)=x0+(j-1)*h, j=1,2,...,N
% with
%   x(N) <= x_end-h,  x(N)+h > x_end-h
% The vector y will contain the estimates of the
% solution Y at the node points in x.
%
n = fix((x_end-x0)/h)+1;
x = linspace(x0,x_end,n)';
y = zeros(n,1);
y(1) = y0;
ft1 = feval(fcn,x(1),y(1));
y(2) = y(1)+h*ft1;
for i = 3:n
  ft2 = feval(fcn,x(i-1),y(i-1));
  y(i) = y(i-1)+h*(3*ft2-ft1)/2;
  ft1 = ft2;
end
```

习题

1. 利用二阶 AB 方法的 MATLAB 程序解 8.5 节习题 2 中的方程. 给出 $h = 0.1$ 和 $h = 0.05$ 时数值解 $y_h(x_n)$ 的理查森误差估计.
2. 修改本节的 MATLAB 程序以便使用三阶 AB 方法. 关于 y_1 和 y_2 的计算, 利用 8.5 节中的二阶龙格–库塔方法. 然后, 重复习题 1. 再继续求解问题 (8.96).

3. 为使 (8.94) 中的误差项更可信一点, 证明

$$\int_0^h g(x)\mathrm{d}x - \frac{h}{2}[3g(0) - g(-h)] = \frac{5}{12}h^3 g''(0) + O(h^4)$$

提示: 把 $g(x)$ 写成在原点的二次泰勒多项式加上误差项 $R_3(x)$. 把它代入到上式的左边, 得到右边.

4. 修改本节的 MATLAB 程序以便使用二阶 AM 方法. 利用二阶 AB 方法作为预估式, 并用欧拉预估式计算 Y_1. 公式 (8.59) 仅仅迭代一次. 把这个方法应用于习题 1 所研究的方程的解, 并给出理查森误差估计.

5. 检查你的计算机上是否已安装有多步亚当斯的程序. 如果有的话, 利用它求解前面习题 1 的方程. 用 $\varepsilon = 10^{-4}$ 和 $\varepsilon = 10^{-8}$ 作为误差容限. 累计所使用的求 $f(x,z)$ 值的次数, 并把它与你自己的亚当斯方法程序所使用的 $f(x,z)$ 求值次数作比较.

6. **(a)** 利用习题 1 的二阶 AB 方法的程序, 对 $0 \leqslant x \leqslant 10$ 解

$$Y'(x) = -50Y(x) + 51\cos(x) + 49\sin(x), \quad Y(0) = 1$$

真解是 $Y(x) = \sin(x) + \cos(x)$. 使用 $h = 0.1, 0.02, 0.01$ 作为步长. 对每种情况, 打印误差以及解答.

(b) 利用习题 4 的二阶 AM 方法的程序, 重复 (a) 中的工作. 检查条件 (8.60).

(c) 当二阶 AM 方法应用于 (a) 中的方程时, y_{n+1} 的值可以直接求出. 执行这样的方法, 重复 (a) 中的工作, 并比较你的结果.

8.7 微分方程组

虽然有些应用只涉及单个的一阶微分方程, 但在很多的应用中, 我们会碰到一阶方程组成的方程组或者高阶方程. 本节探讨一阶方程组的数值解, 指出怎样把前面几节的方法应用于一阶方程组. 至于高阶方程的数值解, 可以先把高阶方程转换成等价的一阶方程组.

从一个简单的案例入手, 先考虑一般形式的两个一阶微分方程组成的方程组

$$\begin{cases} Y_1'(x) = f_1(x, Y_1(x), Y_2(x)) \\ Y_2'(x) = f_2(x, Y_1(x), Y_2(x)) \end{cases} \tag{8.102}$$

函数 $f_1(x, z_1, z_2)$ 和 $f_2(x, z_1, z_2)$ 定义了微分方程, 而未知函数是 $Y_1(x)$ 和 $Y_2(x)$. 初值问题由初始条件

$$Y_1(x_0) = Y_{1,0}, \quad Y_2(x_0) = Y_{2,0} \tag{8.103}$$

和微分方程 (8.102) 组成.

例 8.7.1 (a) 初值问题

$$\begin{cases} Y_1' = Y_1 - 2Y_2 + 4\cos(x) - 2\sin(x), & Y_1(0) = 1 \\ Y_2' = 3Y_1 - 4Y_2 + 5\cos(x) - 5\sin(x), & Y_2(0) = 2 \end{cases} \tag{8.104}$$

有解

$$Y_1(x)=\cos(x)+\sin(x),\quad Y_2(x)=2\cos(x)$$

在后面方程组的欧拉方法的数值例子中将用此例.

(b) 考虑方程组

$$\begin{cases} Y_1'=AY_1[1-BY_2], & Y_1(0)=Y_{1,0} \\ Y_2'=CY_2[DY_1-1], & Y_2(0)=Y_{2,0} \end{cases} \tag{8.105}$$

其中 $A,B,C,D>0$. 这个方程组称为**洛泰卡–沃尔泰拉** (Lotka-Volterra) **食肉动物捕食模型**. 变量 x 表示时间, $Y_1(x)$ 表示在时间 x 被捕食的动物 (如兔子) 个数, 而 $Y_2(x)$ 表示捕食动物 (如狐狸) 个数. 如果只有单独一类捕食动物和单独一类被捕食动物, 则这个模型通常是现实的一个好的近似. 习题 9 讨论解 Y_1 和 Y_2 的性态. ■

对于 m 个一阶微分方程组成的方程组的初值问题, 其一般形式为

$$\begin{cases} Y_1'=f_1(x,Y_1,\cdots,Y_m), & Y_1(x_0)=Y_{1,0} \\ \quad\vdots & \\ Y_m'=f_m(x,Y_1,\cdots,Y_m), & Y_m(x_0)=Y_{m,0} \end{cases} \tag{8.106}$$

我们在某个区间 $x_0\leqslant x\leqslant b$ 上寻找解函数 $Y_1(x),\cdots,Y_m(x)$. (8.120) 中给出由三个方程组成的方程组例子.

通常, 形如 (8.106) 的方程组使用起来不方便, 特别是讨论它的数值解的时候. 方便的做法是利用向量的记号. 定义

$$\boldsymbol{Y}(x)=\begin{bmatrix} Y_1(x) \\ \vdots \\ Y_m(x) \end{bmatrix},\quad \boldsymbol{Y}_0=\begin{bmatrix} Y_{1,0} \\ \vdots \\ Y_{m,0} \end{bmatrix},\quad \boldsymbol{f}(x,z)=\begin{bmatrix} f_1(x,z_1,\cdots,z_m) \\ \vdots \\ f_m(x,z_1,\cdots,z_m) \end{bmatrix} \tag{8.107}$$

其中 $\boldsymbol{z}=[z_1,z_2,\cdots,z_m]^{\mathrm{T}}$. 则 (8.106) 可改写成

$$\boldsymbol{Y}'(x)=\boldsymbol{f}(x,\boldsymbol{Y}(x)),\quad \boldsymbol{Y}(x_0)=\boldsymbol{Y}_0 \tag{8.108}$$

这样一来, 一阶方程组在形式上像前面一直在研究的单个一阶方程. 求解单个一阶方程的数值方法都可以方便地推广到用于解 (8.108) 这样的一阶方程组.

例 8.7.2　方程组 (8.104) 可改写成

$$\boldsymbol{Y}'(x)=\boldsymbol{A}\boldsymbol{Y}(x)+\boldsymbol{G}(x),\quad \boldsymbol{Y}(0)=\boldsymbol{Y}_0$$

其中

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & -2 \\ 3 & -4 \end{bmatrix}$$

$$G(x) = \begin{bmatrix} 4\cos(x) - 2\sin(x) \\ 5\cos(x) - 5\sin(x) \end{bmatrix}, \quad Y_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

用 (8.107) 的记号

$$f(x, z) = Az + G(x), \quad z = [z_1, z_2]^{\mathrm{T}}$$

■

8.7.1 高阶微分方程

在物理学和工程学中, 利用**牛顿第二运动定律**模仿某些自然界中最重要的物理现象导致二阶微分方程. 其他的应用中也能遇到其他一些高阶方程. 高阶方程既能直接地加以研究也能通过等价的一阶方程组加以研究.

例如, 考虑二阶方程

$$Y''(x) = f(x, Y(x), Y'(x)) \tag{8.109}$$

其中 $f(x, z_1, z_2)$ 是给定的. 初值问题由求解从属于初始条件

$$Y(x_0) = Y_0, \quad Y'(x_0) = Y_0' \tag{8.110}$$

的 (8.109) 组成. 为把问题写成一个一阶方程组, 记

$$Y_1(x) = Y(x), \quad Y_2(x) = Y'(x)$$

则 Y_1 和 Y_2 满足

$$\begin{aligned} Y_1' &= Y_2, & Y_1(x_0) &= Y_0 \\ Y_2' &= f(x, Y_1, Y_2), & Y_2(x_0) &= Y_0' \end{aligned} \tag{8.111}$$

从这个方程组出发, 可直接证明 (8.111) 的解 Y_1 也必定满足 (8.109) 和 (8.110), 因而上面这两种形式是等价的.

例 8.7.3 考虑图 8-7 中质量为 m, 摆长为 l 的钟摆. 我们用牛顿第二运动定律导出的二阶微分方程来模拟这个钟摆在其中心线 $\theta = 0$ 附近的运动. 假定钟摆运动时的摩擦力可以忽略不计, 则运动方程为:

$$ml\frac{\mathrm{d}^2\theta}{\mathrm{d}x^2} = -mg\sin(\theta(x)) \tag{8.112}$$

其中 x 是时间, $\theta(x)$ 是垂直中心线和钟摆之间的夹角. 通过指定初始位置 $\theta(0)$ 和初始角速度 $\theta'(0)$ 就完整地描述了钟摆的运动.

为把这个方程转换成两个一阶方程的方程组, 记

$$Y_1(x) = \theta(x), \quad Y_2(x) = \theta'(x)$$

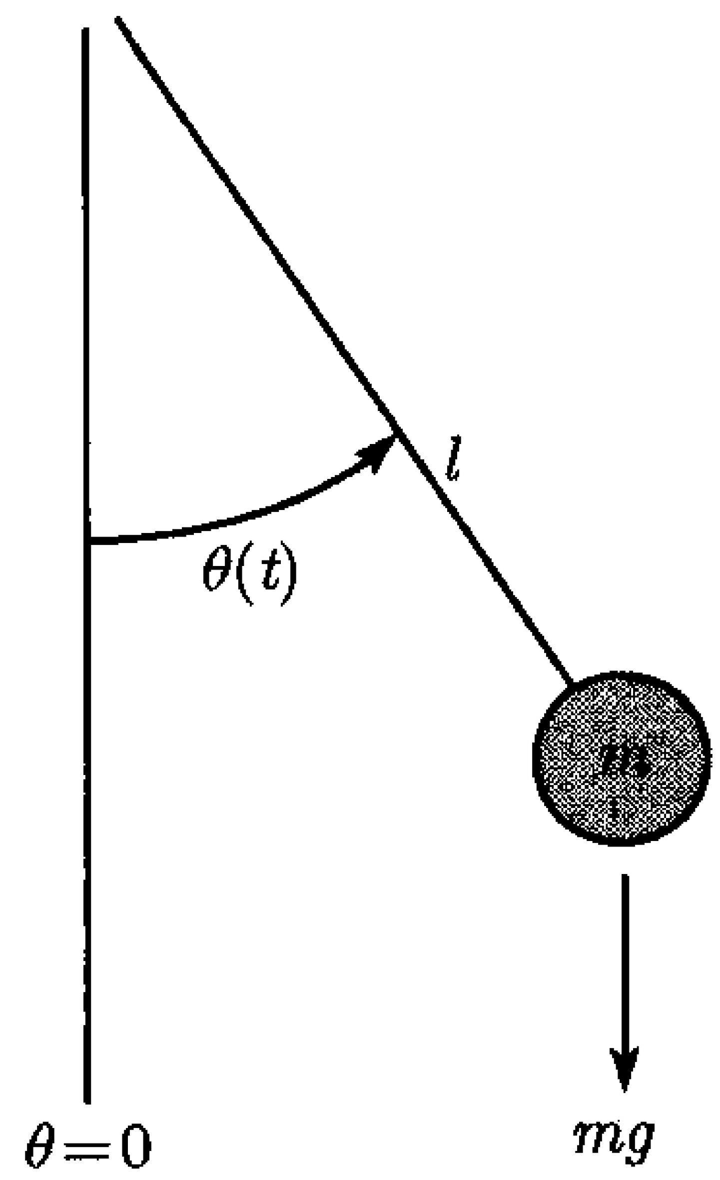


图 8-7 钟摆的图示

则 (8.112) 变成

$$\begin{aligned} Y_1'(x) &= Y_2(x), & Y_1(0) &= \theta(0) \\ Y_2'(x) &= -\frac{g}{l}\sin(Y_1(x)), & Y_2(0) &= \theta'(0) \end{aligned} \tag{8.113}$$

这个方程组等价于原来的二阶方程. ■

一般的 m 阶微分方程可写成

$$\frac{\mathrm{d}^m Y}{\mathrm{d}x^m} = f\left(x, Y, \frac{\mathrm{d}Y}{\mathrm{d}Y}, \cdots, \frac{\mathrm{d}^{m-1}Y}{\mathrm{d}x^{m-1}}\right) \tag{8.114}$$

求解方程所需要的初始条件由下式给出

$$Y(x_0) = y_0, \quad Y'(x_0) = Y_0', \quad \cdots, \quad Y^{(m-1)}(x_0) = Y_0^{(m-1)} \tag{8.115}$$

引进下列变量

$$Y_1 = Y, \quad Y_2 = Y', \quad \cdots, \quad Y_m = Y^{(m-1)}$$

则等价的一阶方程组初值问题为

$$\begin{aligned} Y_1' &= Y_2 & Y_1(x_0) &= Y_0 \\ &\vdots & &\vdots \\ Y_{m-1}' &= Y_m & Y_{m-1}(x_0) &= Y_0^{(m-2)} \\ Y_m' &= f(x, Y_1, \cdots, Y_m) & Y_m(x_0) &= Y_0^{(m-1)} \end{aligned} \tag{8.116}$$

(8.114) 的一个特殊情况是 m 阶线性微分方程

$$\frac{\mathrm{d}^m Y}{\mathrm{d}x^m} = a_0(x)Y + a_1(x)\frac{\mathrm{d}Y}{\mathrm{d}x} + \cdots + a_{m-1}(x)\frac{\mathrm{d}^{m-1}Y}{\mathrm{d}x^{m-1}} + b(x) \tag{8.117}$$

对这个方程, 用

$$Y_m' = a_0(x)Y_1 + a_1(x)Y_2 + \cdots + a_{m-1}(x)Y_m + b(x) \tag{8.118}$$

代替方程组 (8.116) 中最后的那个方程.

例 8.7.4 重组初值问题

$$\begin{cases} Y''' + 3Y'' + 3Y' + Y = -4\sin(x), \\ Y(0) = Y'(0) = 1, \quad Y''(0) = -1 \end{cases} \tag{8.119}$$

为

$$\begin{aligned} Y_1' &= Y_2 & Y_1(0) &= 1 \\ Y_2' &= Y_3 & Y_2(0) &= 1 \\ Y_3' &= -Y_1 - 3Y_2 - 3Y_3 - 4\sin(x) & Y_3(0) &= -1 \end{aligned} \tag{8.120}$$

(8.119) 的解是 $Y(x) = \cos(x) + \sin(x)$, 由它可生成 (8.120) 的解. 在本节后面将数值求解这个方程组. ■

8.7.2 方程组的数值方法

前面几节的数值方法可以几乎不加改变地应用于解一阶微分方程组. 可以对方程组中的每个方程应用数值方法. 或更简单地, 可直接对写成向量形式 (8.108) 的方程组应用数值方法. 推导方程组求解的这些数值方法, 本质上与前面单个方程所做的相同. 而且收敛性和稳定性分析也同样地处理.

作为一个具体的例子, 我们来考虑 (8.102) 中两个一阶方程的一般方程组的欧拉方法. 类似于推导欧拉方法的 (8.27) 式, 泰勒定理给出

$$Y_1(x_{n+1}) = Y_1(x_n) + hf_1(x_n, Y_1(x_n), Y_2(x_n)) + \frac{h^2}{2}Y_1''(\xi_n)$$

$$Y_2(x_{n+1}) = Y_2(x_n) + hf_2(x_n, Y_1(x_n), Y_2(x_n)) + \frac{h^2}{2}Y_2''(\zeta_n)$$

其中 ξ_n 和 ζ_n 在 $[x_n, x_{n+1}]$ 内. 舍弃误差项, 就得到两个一阶方程的方程组的欧拉方法如下: 对 $n \geqslant 0$,

$$\begin{cases} y_{1,n+1} = y_{1,n} + hf_1(x_n, y_{1,n}, y_{2,n}) \\ y_{2,n+1} = y_{2,n} + hf_2(x_n, y_{1,n}, y_{2,n}) \end{cases} \tag{8.121}$$

若 $Y_1(x)$ 和 $Y_2(x)$ 二次连续可微, 则可以证明

$$|Y_1(x_n) - y_{1,n}| \leqslant ch, \quad |Y_2(x_n) - y_{2,n}| \leqslant ch$$

对一切 $x_0 \leqslant x_n \leqslant b$ 和某个常数 c 成立. 此外, 前面的渐近误差公式 (8.38) 依旧成立, 对 $j = 1, 2$

$$Y_j(x_n) - y_{j,n} = D_j(x_n)h + O(h^2), \quad x_0 \leqslant x_n \leqslant b$$

因而, 理查森外推公式和误差估计公式依旧成立. 函数 $D_1(x)$ 和 $D_2(x)$ 满足某个线性微分方程组. 关于单个方程的欧拉方法稳定性结果可直接推广到一阶方程组的情形. 前面关于一阶方程的所有数值方法的研究和讨论都可以类似地推广到一阶方程组的情况, 这也是我们重点讨论解单个的一阶方程的原因.

MATLAB 程序 下面是实施求解初值问题 (8.108) 欧拉方法的 MATLAB 程序 `eulersys`. MATLAB 比 Fortran 优越的一个地方是同样的程序既可用来解单个方程也可用来解方程组. 事实上, 可以看出程序 eulersys 仅少量修改了 8.2 节求解单个方程的程序 `euler_for`. 程序可以自动地确定方程组的方程个数.

```
function [x,y] = eulersys(x0,y0,x_end,h,fcn)
%
% function [x,y]=eulersys(x0,y0,x_end,h,fcn)
%
% Solve the initial value problem of a system
% of first-order equations
%    y' = f(x,y),   x0 <= x <= b,   y(x0)=y0
% Use Euler's method with a stepsize of h.
% The user must supply a program to compute the
% right-hand-side function with some name, say,
% deriv, and a first line of the form
%   function ans=deriv(x,y)
% A sample call would be
%   [t,z]=eulersys(t0,z0,b,delta,'deriv')
%
% The program automatically determines the
% number of equations from the dimension of
% the initial value vector y0.
%
% Output:
% The routine eulersys will return a vector x
% and matrix y. The vector x will contain the
% node points in [x0,x_end]:
%   x(1)=x0, x(j)=x0+(j-1)*h, j=1,2,...,N
% The matrix y is of size N by m, with m the
% number of equations.  The ith row y(i,:) will
% contain the estimates of the solution Y
% at the node points in x(i).
%
m = length(y0);
n = fix((x_end-x0)/h)+1;
x = linspace(x0,x_end,n)';
y = zeros(n,m);
y(1,:) = y0;
```

```
for i = 2:n
  y(i,:) = y(i-1,:) + h*feval(fcn,x(i-1),y(i-1,:));
end
```

例 8.7.5 (a) 利用欧拉方法解 (8.104), 数值解由表 8-22 给出, 其中包括理查森误差估计

$$Y_j(x_n) - y_{j,h}(x_n) \approx y_{j,h}(x_n) - y_{j,2h}(x_n), \quad j = 1,2$$

表 8-22 (8.104) 的欧拉方法解

j	x	$Y_j(x)$	$Y_j(x) - y_{j,2h}(x)$	$Y_j(x) - y_{j,h}(x)$	比率	$y_{j,h}(x) - y_{j,2h}(x)$
	2	0.493 15	−5.65E−2	−2.82E−2	2.0	−2.83E−2
	4	−1.410 45	−5.64E−2	−2.72E−3	2.1	−2.92E−3
1	6	0.680 75	4.81E−2	2.36E−2	2.0	2.44E−2
	8	0.843 86	−3.60E−2	−1.79E−2	2.0	−1.83E−2
	10	−1.383 09	−1.81E−2	−8.87E−3	2.0	−9.40E−2
	2	−0.832 29	−3.36E−2	−1.70E−2	2.0	−1.66E−2
	4	−1.307 29	5.94E−3	3.19E−3	1.9	2.75E−3
2	6	1.920 34	1.59E−3	7.69E−3	2.1	8.17E−3
	8	−0.291 00	−2.08E−2	−1.05E−2	2.0	−1.03E−2
	10	−1.678 14	1.26E−3	9.44E−4	1.3	3.11E−4

在表中, $h = 0.05$, $2h = 0.1$. 可以看出这个误差估计颇为精确, 除了 $j = 2$, $x = 10$ 时的情况. 为得到给定节点 $x = 2, 4, 6, 8, 10$ 处的数值解值及其误差, 使用下列 MATLAB 指令, 对这个例子而言, 指令可以包括在程序 eulersys的末端.

```
n1 = (n-1)/5;
for i = n1+1:n1:n
  e(i,1) = cos(x(i))+sin(x(i))-y(i,1);
  e(i,2) = 2*cos(x(i))-y(i,2);
end
diary euler_sys1
fprintf(' h = 6.5f\n', h)
disp(' x y(1) e(1) y(2) e(2)')
for i = n1+1:n1:n
  fprintf('2.0f%10.2e%10.2e%10.2e%10.2e\n', ...
      x(i), y(i,1),e(i,1),y(i,2),e(i,2))
end
diary off
```

本例的右端函数由下列程序定义:

```
function z = eulersys_fcn(x,y);
z = zeros(1,2);
z(1) = y(1)-2*y(2)+4*cos(x)-2*sin(x);
z(2) = 3*y(1)-4*y(2)+5*cos(x)-5*sin(x);
```

(b) 利用解重组问题 (8.120) 的欧拉方法, 解 (8.119) 的三阶方程. 表 8-23 对步

长 $2h=0.1$ 和 $h=0.05$ 给出 $y(x)=Y_1(x)=\sin(x)+\cos(x)$ 的结果. 理查森误差估计还是颇为准确. ■

表 8-23 (8.119) 的欧拉方法解

x	$y(x)$	$y(x)-y_{2h}(x)$	$y(x)-y_h(x)$	比率	$y_h(x)-y_{2h}(x)$
2	0.493 15	−8.78E−2	−4.25E−2	2.1	−4.53E−2
4	−1.410 45	1.39E−1	6.86E−2	2.0	7.05E−2
6	0.680 75	5.19E−2	2.49E−2	2.1	2.70E−2
8	0.843 86	−1.56E−1	−7.56E−2	2.1	−7.99E−2
10	−1.383 09	8.39E−2	4.14E−2	2.0	4.25E−2

其他的数值方法也可以同样直接地应用于方程组. 并且对向量形式的方程组 (8.108), 数值方法的形式是一样的. 例如, 龙格–库塔方法 (8.81) 是

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_n+hf(x_n,y_n))],\quad n\geqslant 0 \tag{8.122}$$

对两个方程的方程组用

$$\boldsymbol{y}_n=\begin{bmatrix} y_{1,n}\\ y_{2,n}\end{bmatrix},\quad \boldsymbol{f}(x_n,\boldsymbol{y}_n)=\begin{bmatrix} f_1(x_n,y_{1,n},y_{2,n})\\ f_2(x_n,y_{1,n},y_{2,n})\end{bmatrix}$$

解释这个公式时. 对 $j=1,2$, 方法为

$$\begin{aligned} y_{j,n+1}=&y_{j,n}+\frac{h}{2}[f_j(x_n,y_{1,n},y_{2,n})\\ &+f_j(x_{n+1},y_{1,n}+hf_1(x_n,y_{1,n},y_{2,n}),y_{2,n}+hf_2(x_n,y_{1,n},y_{2,n}))]\end{aligned} \tag{8.123}$$

以 (8.122) 形式考虑这个方法比较简单, 在计算机上对它编程, 向量形式十分方便. 我们把 (8.123) 的说明留作习题.

习题

1. 设

$$\boldsymbol{A}=\begin{bmatrix} 1 & -2\\ 2 & -1\end{bmatrix},\quad \boldsymbol{Y}=\begin{bmatrix} Y_1\\ Y_2\end{bmatrix}$$

$$\boldsymbol{G}(x)=\begin{bmatrix} -2\mathrm{e}^{-x}+2\\ -2\mathrm{e}^{-x}+1\end{bmatrix},\quad \boldsymbol{Y}_0=\begin{bmatrix} 1\\ 1\end{bmatrix}$$

写出组成方程组

$$\boldsymbol{Y}'=\boldsymbol{A}\boldsymbol{Y}+\boldsymbol{G}(x),\quad \boldsymbol{Y}(x_0)=\boldsymbol{Y}_0$$

的两个方程. 真解是 $\boldsymbol{Y}=[\mathrm{e}^{-x},1]^{\mathrm{T}}$.

2. 把方程组 (8.120) 转换成习题 1 中那样的一般形式, 给出矩阵 $\boldsymbol{A}$.

3. 把下列高阶方程转换成一阶方程组:

(a) $Y'''+4Y''+5Y'+2Y=2x^2+10x+8$

$Y(0)=1,\quad Y'(0)=-1,\quad Y''(0)=3$

真解是 $Y(x)=\mathrm{e}^{-x}+x^2$.

(b) $Y''+4Y'+13Y=40\cos(x),\quad Y(0)=3,\quad Y'(0)=4$

真解是 $Y(x)=3\cos(x)+\sin(x)+\mathrm{e}^{-2x}\sin(3x)$.

4. 把下列二阶方程组

$$x''(t)=\frac{-cx(t)}{r(t)^3},\quad y''(t)=\frac{-cy(t)}{r(t)^3},\quad z''(t)=\frac{-cz(t)}{r(t)^3}$$

转换成一个较大的一阶方程组. 它出自研究一个质点对另一个质点的万有引力吸引力, 这里 c 为正常数, $r(t)=\left[x(t)^2+y(t)^2+z(t)^2\right]^{1/2}$, t 代表时间.

5. 利用欧拉方法解习题 1 中的方程组. 利用步长 $h=0.1, 0.05, 0.025$, 并对 $0\leqslant x\leqslant 10$ 求解. 对 $h=0.025$ 利用理查森误差公式估计误差.

6. 利用向后欧拉方法重复习题 5.

7. 对习题 3 中的方法组重复习题 5.

8. 修改本节的欧拉程序, 实施 (8.123) 中给出的龙格–库塔方法. 用这个程序重复习题 5 和习题 7.

9. 考虑 (8.105) 的食肉动物捕食模型, 其中的常数取值如下: $A=4, B=0.5, C=3, D=\frac{1}{3}$.

(a) 证明存在解 $Y_1(x)=C_1$, $Y_2(x)=C_2$, 其中 C_1 和 C_2 为常数. 这样的解 $Y(x)$ 的物理意义是什么?

(b) 对 $0\leqslant x\leqslant 4$ 用 $Y_1(0)=3$, $Y_2(0)=5$ 求解方程组 (8.105), 利用习题 8 的龙格–库塔方法, 步长 $h=0.01$ 和 0.005. 考察并在 x 方向以步长 0.1 画出输出值. 此外画出 $Y_1(x)$ 对 x 以及 $Y_2(x)$ 对 x 的值, 再画出 Y_1 对 Y_2 的值.

(c) 依次对初值 $Y_1(0)=3$, $Y_2(0)=1, 1.5, 1.9$ 重复 (b). 说明这些解与别的解的关系以及与 (a) 的解的关系.

10. 考虑求解钟摆方程 (8.112), 其中 $l=1$ 英尺, $g=32.2$ 英尺/秒 ^{2}. 关于初值, 选择 $0<\theta(0)\leqslant\pi/2$, $\theta'(0)=0$. 利用欧拉方法求解 (8.113), 为了得到计算解中适当小的误差, 用各种 h 值进行试验. 画出 t 对 $\theta(t)$, t 对 $\theta'(t)$ 以及 $\theta(t)$ 对 $\theta'(t)$ 的图形. 钟摆运动关于时间是周期变化的吗?

8.8 两点边值问题的有限差分法

前节中已经看到, 二阶方程

$$Y''=f(x,Y,Y') \tag{8.124}$$

的初值问题可以重整为一个一阶方程组的初值问题. 本节我们考虑二阶方程 (8.124) 另一类问题的解, 该问题在两个不同的 x 值处给出解 Y 的条件. 这种问题称为***两点边值问题***. 为简单起见, 本节将专门讨论下列二阶线性方程的边值问题:

$$\begin{cases} Y''(x)=p(x)Y'(x)+q(x)Y(x)+r(x), & a\leqslant x\leqslant b \\ Y(a)=g_1, \quad Y(b)=g_2 \end{cases} \tag{8.125}$$

条件 $Y(a)=g_1$ 和 $Y(b)=g_2$ 称为**边界条件**. 包含未知函数导数的边界条件在应用中也很常见, 本节后面将讨论这种边界条件的有限差分逼近.

假定给定的函数 p, q 和 r 在 $[a,b]$ 上连续. 根据相关的数学理论, 若对 $x\in[a,b]$, $q(x)>0$, 则边值问题 (8.125) 有唯一解. 我们将假定问题有唯一的光滑解 Y.

有限差分方法的主要特征是用适当的有限差分代替导数得到离散方程. 对边值问题 (8.125), 用三步导出有限差分方程组.

第一步, 将区间 $[a,b]$ 离散. 设 N 为正整数, 把区间 $[a,b]$ 分成 N 个相等的部分:

$$[a,b]=[x_0,x_1]\cup[x_1,x_2]\cup\cdots\cup[x_{N-1},x_N]$$

其中 $a=x_0<x_1<\cdots<x_{N-1}<x_N=b$ 是网格点 (或结点). 记 $h=(b-a)/N$, 称为**步长**. 然后用

$$x_i=a+ih,\quad 0\leqslant i\leqslant N$$

给出结点. 区间的非均匀划分也是可能的, 事实上, 当边值问题 (8.125) 的解在区间的某个部分比其余部分更加迅速地改变时, 非均匀划分更好. 为简单起见, 这里仅限于讨论均匀划分情况. 引进记号 $p_i=p(x_i)$, $q_i=q(x_i)$, $r_i=r(x_i)$, $0\leqslant i\leqslant N$, 并用 $y_i(0\leqslant i\leqslant N)$ 表示真解 $Y_i=Y(x_i)(0\leqslant i\leqslant N)$ 的数值近似值.

第二步, 在内部结点 $x_1,\cdots,x_{N-1}$ 处离散微分方程. 为此, 回顾下列差分逼近公式 (参见 (5.85) 和 (5.92)):

$$Y'(x_i)=\frac{Y_{i+1}-Y_{i-1}}{2h}+O(h^2) \tag{8.126}$$

$$Y''(x_i)=\frac{Y_{i+1}-2Y_i+Y_{i-1}}{h^2}+O(h^2) \tag{8.127}$$

则在 $x=x_i$ 处的微分方程变成

$$\frac{Y_{i+1}-2Y_i+Y_{i-1}}{h^2}=p_i\frac{Y_{i+1}-Y_{i-1}}{2h}+q_iY_i+r_i+O(h^2) \tag{8.128}$$

舍弃余项 $O(h^2)$, 并用 y_i 代替 Y_i, 得到差分方程

$$\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}=p_i\frac{y_{i+1}-y_{i-1}}{2h}+q_iy_i+r_i,\quad 1\leqslant i\leqslant N-1$$

它可改写为

$$-\left(1+\frac{h}{2}p_i\right)y_{i-1}+(2+h^2q_i)y_i+\left(\frac{h}{2}p_i-1\right)y_{i+1}=-h^2r_i,\quad 1\leqslant i\leqslant N-1 \tag{8.129}$$

第三步, 边界条件的处理. 差分方程 (8.129) 给出了关于 $(N+1)$ 个未知量 $y_0,y_1,\cdots,y_N$ 的 $(N-1)$ 个方程. 我们需要补充两个方程, 这两个方程来自边界条件的离散化. 对模型问题 (8.125), 边界条件的离散化是简单的:

$$y_0=g_1,\quad y_N=g_2 \tag{8.130}$$

方程 (8.129) 和 (8.130) 一起构成一个线性方程组. 因为 y_0 和 y_N 的值在 (8.130) 中明确地给出, 所以可以从线性方程组中消去 y_0 和 y_N. 用 $y_0 = g_1$, 可改写 (8.129) 中 $i=1$ 的方程为

$$(2+h^2q_1)y_1 + \left(\frac{h}{2}p_1 - 1\right)y_2 = -h^2r_1 + \left(1+\frac{h}{2}p_1\right)g_1 \tag{8.131}$$

类似地, 从 (8.129) 中 $i = N-1$ 的方程, 得到

$$-\left(1+\frac{h}{2}p_{N-1}\right)y_{N-2} + (2+h^2q_{N-1})y_{N-1} = -h^2r_{N-1} + \left(1-\frac{h}{2}p_{N-1}\right)g_2 \tag{8.132}$$

所以最后, 未知的数值解向量 $\boldsymbol{y} = [y_1, \cdots, y_{N-1}]^{\mathrm{T}}$ 的有限差分方程组为

$$\boldsymbol{Ay} = \boldsymbol{b} \tag{8.133}$$

其中

$$\boldsymbol{A} = \begin{bmatrix} 2+h^2q_1 & \frac{h}{2}p_1 - 1 & & & \\ -\left(1+\frac{h}{2}p_2\right) & 2+h^2q_2 & \frac{h}{2}p_2 - 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\left(1+\frac{h}{2}p_{N-2}\right) & 2+h^2q_{N-2} & \frac{h}{2}p_{N-2}-1 \\ & & & -\left(1+\frac{h}{2}p_{N-1}\right) & 2+h^2q_{N-1} \end{bmatrix} \tag{8.134}$$

是系数阵, 而

$$\boldsymbol{b} = \left[-h^2r_1 + \left(1+\frac{h}{2}p_1\right)g_1, -h^2r_2, \cdots, -h^2r_{N-2}, -h^2r_{N-1} + \left(1-\frac{h}{2}p_{N-1}\right)g_2\right]^{\mathrm{T}} \tag{8.135}$$

是右端向量.

可以证明, 若真解 $Y(x)$ 充分光滑, 比如说它有直到 4 阶的连续导数, 则差分格式 (8.133) 是二阶方法, 即

$$\max_{0\leqslant i\leqslant N} |Y(x_i) - y_i| = O(h^2)$$

而且对某个与 h 无关的函数 $D(x)$, 下列渐近误差展开

$$Y(x_i) - y_h(x_i) = h^2D(x_i) + O(h^4), \quad 0 \leqslant i \leqslant N \tag{8.136}$$

成立. 这时的理查森外推公式是

$$\tilde{y}_h(x_i) = \frac{4y_h(x_i) - y_{2h}(x_i)}{3} \tag{8.137}$$

而且有

$$Y(x_i) - \tilde{y}_h(x_i) = O(h^4) \tag{8.138}$$

线性方程组 (8.133) 是三对角的. 故可使用 6.4 节讨论的算法求解方程组 (8.133).

MATLAB 程序　下列 MATLAB 代码实施求解问题 (8.125) 的差分方法 (8.133):

```
function z = ODEBVP(p,q,r,a,b,ga,gb,N)
% A program to solve the two-point boundary
% value problem
%   y"=p(x)y'+q(x)y+r(x),  a<x<b
%   y(a)=ga,  y(b)=gb
% Input
%   p, q, r: coefficient functions
%   a, b: the end-points of the interval
%   ga, gb: the prescribed function values
%           at the end-points
%   N: number of subintervals
% Output
%   z = [ xx yy ]: xx is an (N+1) column vector
%                  of the node points
%                 yy is an (N+1) column vector of
%                  the solution values
% A sample call would be
%   z=ODEBVP('p','q','r',a,b,ga,gb,100)
% The user must provide m-files to define the
% functions p, q and r.
%
% Other MATLAB program called: tridiag.m
%
% Initialization
N1 = N+1;
h = (b-a)/N;
h2 = h*h;
xx = linspace(a,b,N1)';
yy = zeros(N1,1);
yy(1) = ga;
yy(N1) = gb;
% Define the subdiagonal avec, main diagonal bvec,
% superdiagonal cvec
pp(2:N) = feval(p,xx(2:N));
avec(2:N-1) = -1-(h/2)*pp(3:N);
bvec(1:N-1) = 2+h2*feval(q,xx(2:N));
cvec(1:N-2) = -1+(h/2)*pp(2:N-1);
% Define the right-hand-side vector fvec
fvec(1:N-1) = -h2*feval(r,xx(2:N));
```

```
fvec(1) = fvec(1)+(1+h*pp(2)/2)*ga;
fvec(N-1) = fvec(N-1)+(1-h*pp(N)/2)*gb;
% Solve the tridiagonal system
yy(2:N) = tridiag(avec,bvec,cvec,fvec,N-1,0);
z = [xx'; yy']';
```

例 8.8.1 考虑边值问题

$$\begin{cases} Y'' = -\dfrac{2x}{1+x^2}Y' + Y + \dfrac{2}{1+x^2} - \log(1+x^2), \quad 0 \leqslant x \leqslant 1 \\ Y(0) = 0, \quad Y(1) = \log(2) \end{cases} \tag{8.139}$$

真解是 $Y(x) = \ln(1+x^2)$.

表 8-24 对几个 h 值在选择的结点处给出有限差分解的误差 $Y - y_h$. 表 8-25 在同样的结点处给出外推解的误差 $Y - (4y_h - y_{2h})/3$ 以及不同步长相关误差的比. 步长为 h 的解的误差列近旁有标记为"比率"的列, 其中的数字是步长为 h 的解误差与步长为 $2h$ 的解误差之比. 我们看到当步长折半时误差大约减少到原来的 1/4, 表明方法的 2 阶收敛性. 通过外推, 数值解的精度有很大的改进. $h = 1/40$ 时的外推解 $\tilde{y}_h$ 比 $h = 1/160$ 时的解 y_h 更加准确. 注意到计算 $h = 1/40$ 时的 $\tilde{y}_h$ 的工作

表 8-24 解问题 (8.139) 的数值误差 $Y(x) - y_h(x)$

x	$h=1/20$	$h=1/40$	比率	$h=1/80$	比率	$h=1/160$	比率
0.1	5.10E−5	1.27E−5	4.00	3.18E−6	4.00	7.96E−7	4.00
0.2	7.84E−5	1.96E−5	4.00	4.90E−6	4.00	1.22E−6	4.00
0.3	8.64E−5	2.16E−5	4.00	5.40E−6	4.00	1.35E−6	4.00
0.4	8.08E−5	2.02E−5	4.00	5.05E−6	4.00	1.26E−6	4.00
0.5	6.73E−5	1.68E−5	4.00	4.21E−6	4.00	1.05E−6	4.00
0.6	5.08E−5	1.27E−5	4.00	3.17E−6	4.00	7.94E−7	4.00
0.7	3.44E−5	8.60E−6	4.00	2.15E−6	4.00	5.38E−7	4.00
0.8	2.00E−5	5.01E−6	4.00	1.25E−6	4.00	3.13E−7	4.00
0.9	8.50E−6	2.13E−6	4.00	5.32E−7	4.00	1.33E−7	4.00

表 8-25 解问题 (8.139) 的外推误差

x	$h=1/40$	$h=1/80$	比率	$h=1/160$	比率
0.1	−9.23E−09	−5.76E−10	16.01	−3.60E−11	16.00
0.2	−1.04E−08	−6.53E−10	15.99	−4.08E−11	15.99
0.3	−6.60E−09	−4.14E−10	15.96	−2.59E−11	15.98
0.4	−1.18E−09	−7.57E−11	15.64	−4.78E−12	15.85
0.5	3.31E−09	2.05E−10	16.14	1.28E−11	16.06
0.6	5.76E−09	3.59E−10	16.07	2.24E−11	16.04
0.7	6.12E−09	3.81E−10	16.04	2.38E−11	16.03
0.8	4.88E−09	3.04E−10	16.03	1.90E−11	16.03
0.9	2.67E−09	1.67E−10	16.02	1.04E−11	16.03

量远小于计算 $h=1/160$ 时的 y_h 的工作量.

再观察外推解 $\tilde{y}_h$, 当步长折半时误差大约减少到原来的 1/16. 实际上可以证明, 若真解 $Y(x)$ 六次连续可微, 则可改进渐近误差展开 (8.136) 为

$$Y(x_i)-y_h(x_i)=h^2D_1(x_i)+h^4D_2(x_i)+O(h^6)$$

然后用

$$Y(x_i)-\tilde{y}_h(x_i)=-4h^4D_2(x_i)+O(h^6)$$

替代 (8.137). 所以我们能用公式

$$Y(x_i)-\frac{16\tilde{y}_h(x_i)-\tilde{y}_{2h}(x_i)}{15}=O(h^6)$$

对 $\tilde{y}_h$ 执行外推过程得到更为准确的数值解. 例如, 在 $x_i=0.5$ 处, 用 $h=1/80$, 进一步的外推解的误差约等于 $-1.88\text{E}-12$. ■

至今, 讨论集中在解特殊的边值问题 (8.125). 下面讨论各种推广.

原则上, 可以类似地导出求解更一般的方程 (8.124) 的差分格式. 例如, 若使用公式 (8.126) 和 (8.127), 在内部结点 x_i 处可以通过差分方程

$$\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}=f\left(x_i,y_i,\frac{y_{i+1}-y_{i-1}}{2h}\right)$$

逼近方程 (8.124).

包含未知量导数的边界条件的处理有点复杂. 假定在 $x=b$ 处的边界条件用

$$Y'(b)+kY(b)=g_2 \tag{8.140}$$

代替. 一个显然的可能是用 $(Y_N-Y_{N-1})/h$ 逼近 $Y'(b)$. 然而

$$Y'(b)-\frac{Y_N-Y_{N-1}}{h}=O(h)$$

这个逼近的精度比 (8.128) 中的余项 $O(h^2)$ 低一阶. 结果, 下列离散的边界条件

$$\frac{y_N-y_{N-1}}{h}+ky_N=g_2 \tag{8.141}$$

对应的差分解只有 $O(h)$ 的精度. 为保持差分解的二阶收敛, 我们需要更精确地逼近边界条件 (8.140). 一种处理方法是基于公式

$$Y'(b)=\frac{3Y_N-4Y_{N-1}+Y_{N-2}}{2h}+O(h^2) \tag{8.142}$$

然后用

$$\frac{3y_N-4y_{N-1}+y_{N-2}}{2h}+ky_N=g_2 \tag{8.143}$$

来逼近边界条件 (8.140). 可以证明, 所得的差分格式具有二阶精度. 有其他的可能逼近边界条件 (8.140), 使整体差分格式保持二阶精度, 例如, 9.3 节中通过利用新的变量数值上处理导数初值条件 (参见 (9.32) 以及那里的讨论).

习题

1. 一般来说, 研究边值问题解的存在性和唯一性要比研究初值问题的复杂. 考虑边值问题

$$\begin{cases} Y''=0, & 0<x<1 \\ Y'(0)=g_1, & Y'(1)=g_2 \end{cases}$$

证明: 当 $g_1 \neq g_2$ 时问题无解, 而当 $g_1 = g_2$ 时问题有无限多解.

提示: 对 $g_1 \neq g_2$ 情况, 在 $[0,1]$ 上积分微分方程.

2. 作为解不唯一的另一个例子, 对任意的常数 c, 证明 $Y(x)=c\sin(x)$ 是边值问题

$$\begin{cases} Y''(x)+Y(x)=0, & 0<x<\pi \\ Y(0)=Y(\pi)=0 \end{cases}$$

的解.

3. 证明: 任何形如 $Y(x)=c_1\mathrm{e}^x+c_2\mathrm{e}^{-x}$ 的函数满足方程

$$Y''(x)-Y(x)=0$$

对满足下列边界条件的函数 $Y(x)$ 确定 c_1 和 c_2:

(a) $Y(0)=1, \quad Y(1)=0$;

(b) $Y(0)=1, \quad Y'(1)=0$;

(c) $Y'(0)=1, \quad Y(1)=0$.

4. 假定 Y 三次连续可微. 利用泰勒定理证明公式 (8.142).

5. 利用渐近展开 (8.136) 证明公式 (8.138).

6. 证明: 任何形如 $Y(x)=c_1\sqrt{x}+c_2x^4$ 的函数满足方程

$$x^2Y''(x)-\frac{7}{2}xY'(x)+2Y(x)=0$$

用边界条件

$$Y(1)=1, \quad Y(4)=2$$

确定方程的解. 对 $h=0.1, 0.05, 0.025$ 利用 MATLAB 程序 ODEBVP 求解边值问题, 在 $x=1.2, 1.4, 1.6, 1.8$ 处打印数值解的误差. 评述 h 折半时误差怎样减少. 对外推解做同样的工作.

7. 方程

$$x^2Y''-x(x+2)Y'+(x+2)Y=0$$

的通解是 $Y(x)=c_1x+c_2x\mathrm{e}^x$. 用边界条件

$$Y(1)=\mathrm{e}, \quad Y(2)=2\mathrm{e}^2$$

确定方程的一个特解. 对 $h=0.1, 0.05, 0.025$ 利用 MATLAB 程序 ODEBVP 求解边值问题, 在 $x=1.2, 1.4, 1.6$ 和 1.8 处打印数值解的误差. 评述当 h 折半时误差怎样减少. 对外推的解做同样的工作.

8. 方程

$$xY'' - (2x+1)Y' + (x+1)Y = 0$$

的通解是 $Y(x) = c_1 e^x + c_2 x^2 e^x$. 用边界条件

$$Y'(1) = 0, \quad Y(2) = e^2$$

求方程的一个特解. 对边界条件 $Y'(1) = 0$ 的离散逼近给出一个精度为 $O(h^2)$ 的类似于 (8.143) 的公式. 修改程序 ODEBVP 以实施你的方法, 并对 $h = 0.1, 0.05, 0.025$ 求解问题. 打印数值解在 $x = 1, 1.2, 1.4, 1.6, 1.8$ 处的误差, 评述 h 折半时误差怎样减少. 对外推的解做同样的工作.

9. 对常数 p, q 和 r 考虑边值问题 (8.125). 修改 MATLAB 程序使得指令 `feval` 不出现. 利用修改的程序求解下列边值问题:

(a) $$\begin{cases} Y'' = -Y, \quad 0 < x < \dfrac{\pi}{2} \\ Y(0) = Y\left(\dfrac{\pi}{2}\right) = 1 \end{cases}$$

真解是 $Y(x) = \sin(x) + \cos(x)$.

(b) $$\begin{cases} Y'' + Y = \sin(x), \quad 0 < x < \dfrac{\pi}{2} \\ Y(0) = Y\left(\dfrac{\pi}{2}\right) = 0 \end{cases}$$

真解是 $Y(x) = -\frac{x}{2}\cos x$.

10. 对下列边值问题

$$\begin{cases} Y'' = \sin(xY') + 1, \quad 0 < x < 1 \\ Y(0) = 0, \quad Y(1) = 1 \end{cases}$$

给出一个二阶格式.

第 9 章　偏微分方程的有限差分法

科学和工程学中的许多现象依赖于多个独立变量. 例如, 现实世界的未知函数通常同时依赖于时间 t 和点 (x, y, z) 的位置. 因而这个未知函数的微分方程包含函数关于这些独立变量的偏导数. 由此得到偏微分方程. 两个独立变量的偏微分方程的一些例子是

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \quad (x, y) \in \Omega \subset \mathbb{R}^2 \tag{9.1}$$

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + f(x, t), \quad x \in (0, L), t > 0 \tag{9.2}$$

$$\frac{\partial^2 u}{\partial t^2} = a\frac{\partial^2 u}{\partial x^2} + f(x, t), \quad x \in (0, L), t > 0 \tag{9.3}$$

其中 Ω 是 R^2 中的一个区域, 常数 $L > 0$, 系数 $a > 0$, 而函数 $f(x, y)$ 或 $f(x, t)$ 是给定的. 这些方程分别称为*泊松方程*、*热传导方程*和*波动方程*. 齐次泊松方程, 即 $f(x, y) = 0$ 时的方程 (9.1) 称为*拉普拉斯*(Laplace)*方程*.

泊松方程可用作区域 Ω 内温度的稳态分布模型, 这里把 u 看作温度而把 $-f$ 看作 (标度的) 热源分布. 它也可作为以 $f(x, y)$ 代表电荷密度的静电势模型. 拉普拉斯方程可作为某些二维流体流动形式的模型. 热传导方程可用于描述沿着一根细杆的热转移过程, 系数 a 由热传导系数、具体的热度和密度确定的. 这个方程也能作为扩散过程的模型, 所以也称为*扩散方程*. 波动方程可用作弦的振动模型, 系数 a 与弦的张力和密度有关. 量 $\sqrt{a}$ 可看作波传播的速度. 方程的二维形式, 其中右端项 $\frac{\partial^2 u}{\partial x^2}$ 用 $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ 代替, 可作为弹簧膜振动的模型.

泊松方程是*椭圆型方程*的一个典型的模型, 而热传导方程和波动方程分别是*抛物型方程*和*双曲型方程*. 一般地, 对于某些常数 A, B, C, 以及函数 $F(x, y, u, p, q)$, 考虑下列形式的两个独立变量的二阶偏微分方程

$$Au_{xx} + Bu_{xy} + Cu_{yy} = F(x, y, u, u_x, u_y) \tag{9.4}$$

用 $\Delta = B^2 - 4AC$ 表示判别式. 若 $\Delta < 0$ 则方程为椭圆型, 若 $\Delta = 0$ 则方程为抛物型, 若 $\Delta > 0$ 则方程为双曲型. 把微分方程分类是有用的, 因为不同类型的微分方程有不同的解性质, 而这些性质对选择和执行数值方法有重要的影响. 对于来自各种应用的抛物型和双曲型方程, 可以把两个独立变量之一看作时间. 因此, 抛物型和双曲型方程将使用 x 和 t 作为独立变量.

泊松方程 (9.1) 具有 (9.4) 的形式, 其中 $A=C=1$ 而 $B=0$. 它的判别式 $\Delta=0^2-4\times1\times1=-4<0$. 故泊松方程是椭圆型的. 热传导方程 (9.2) 可改写为

$$a\frac{\partial^2 u}{\partial x^2}=\frac{\partial u}{\partial t}-f(x,t)$$

而且可取 $A=a$, $B=C=0$. 于是判别式为 0, 方程是抛物型的. 最后把波动方程 (9.3) 写成 (9.4) 的形式:

$$a\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial t^2}=-f(x,t)$$

于是, $A=a$, $B=0$ 而 $C=-1$, 判别式 $\Delta=4a>0$ 且方程是双曲型的. 请读者确定下列方程的类型:

$$4\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y)$$

$$4\frac{\partial^2 u}{\partial x^2}+4\frac{\partial^2 u}{\partial x\partial y}+\frac{\partial^2 u}{\partial y^2}=f(x,y)$$

$$4\frac{\partial^2 u}{\partial x^2}-8\frac{\partial^2 u}{\partial x\partial y}+\frac{\partial^2 u}{\partial y^2}=f(x,y)$$

偏微分方程求解比常微分方程求解更困难. 偏微分方程的解, 指的是一个满足微分方程的函数. 为了唯一地确定一个解, 通常需要指定某些称为边值条件或者初值条件的补充条件. 这样构成一个偏微分方程问题: 初值问题、边值问题或者初边值问题. 一个基本的事实是一般不可能找到偏微分方程解的公式. 应用中产生的偏微分方程通常用数值方法求解. 在各种数值方法中, 有限差分法是容易推导和实施的一种方法. 本章中将研究求解模型偏微分方程 (9.1)~(9.3) 的有限差分法.

9.1　泊 松 方 程

设 $\Omega\subset R^2$ 是一个平面区域, 用 $\partial\Omega$ 表示它的边界. 本节中考虑的边值问题是

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}&=f(x,y),\quad (x,y)\in\Omega\\ u&=g(x,y),\quad (x,y)\in\partial\Omega\end{aligned}\tag{9.5}$$

指定边界上未知解的值. 这样的条件称为*狄利克雷* (Dirichlet)*边界条件*. 对 2 阶微分方程, 也可以指定包含一阶偏导数的边界条件, 这产生了*诺伊曼* (Neumann)*边界条件*或*罗宾* (Robin)*边界条件*. 下面只考虑狄利克雷边界条件.

为讨论简单起见, 假定 Ω 是正方形: $\Omega=(0,1)\times(0,1)$. 于是边界 $\partial\Omega$ 由四条线段组成, 它们是正方形的四条边. 下面讨论问题的有限差分法. 把 x 的区间 $[0,1]$

分成 n_x 等分并用 $h_x = 1/n_x$ 表示 x 的步长. 类似地, 把 y 的区间 $[0,1]$ 分成 n_y 等分并用 $h_y = 1/n_y$ 表示 y 的步长. 于是网格点是

$$(x_i, y_i), 1 \leqslant i \leqslant n_x+1, 1 \leqslant j \leqslant n_y+1$$

其中 $x_i = (i-1)h_x$, $y_i = (j-1)h_y$. 这样的有限差分网格见图 9-1. 注意, 传统上按照

$$(x_i, y_i), 0 \leqslant i \leqslant n_x, 0 \leqslant j \leqslant n_y$$

对网格点编号. 我们选择的编号是为了能更容易在 MATLAB 中实施数值格式.

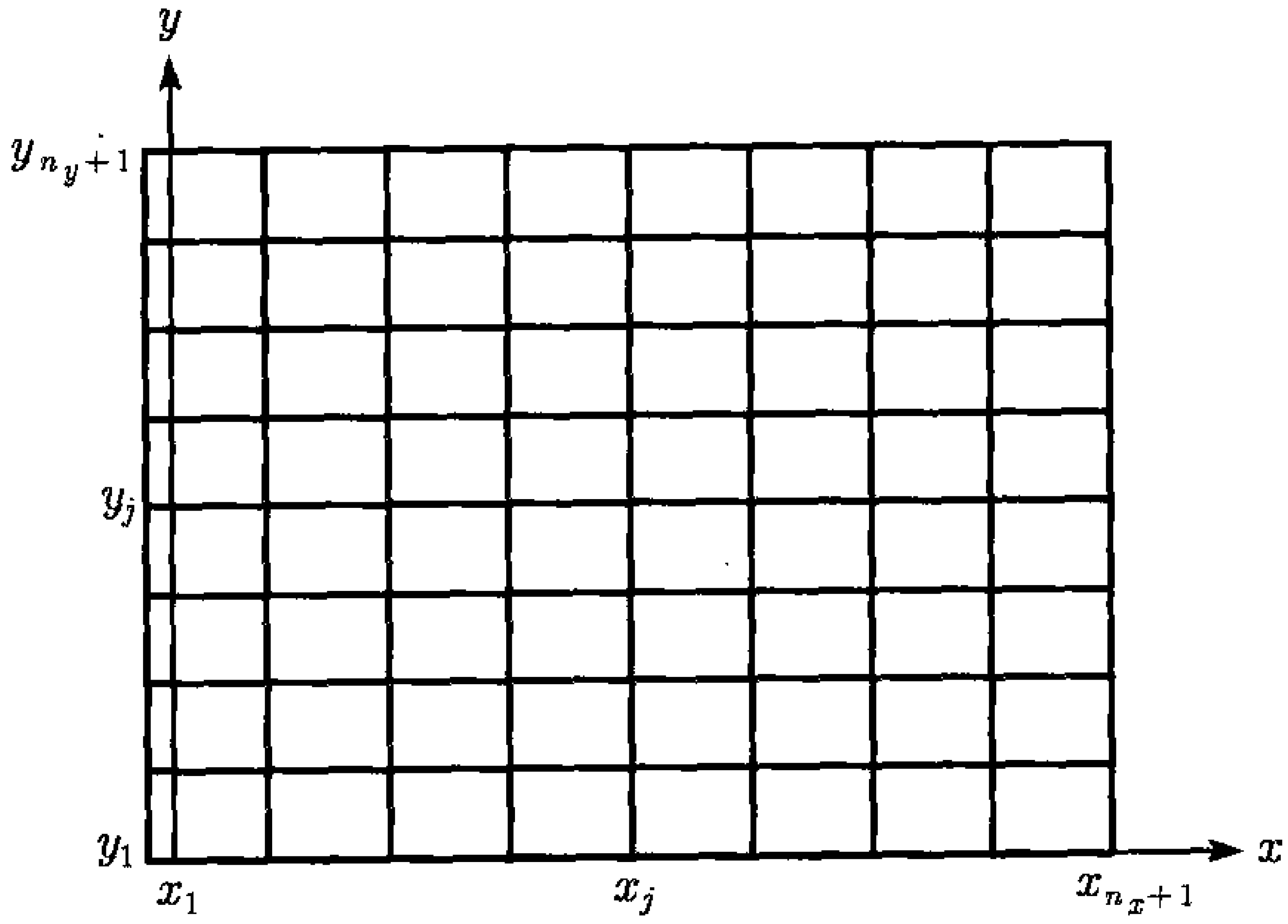


图 9-1 有限差分网格

考虑内部网格点 (x_i, y_j) 处的微分方程, 其中 $2 \leqslant i \leqslant n_x, 2 \leqslant j \leqslant n_y$. 利用三点中心差分逼近二阶导数:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h_x^2} + O(h_x^2)$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{h_y^2} + O(h_y^2)$$

这样, 在 (x_i, y_j) 处的微分方程可近似地写成

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h_x^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{h_y^2} \approx f_{ij} \tag{9.6}$$

其中 $f_{ij} = f(x_i, y_j)$. 用 u_{ij} 表示 $u(x_i, y_j)$ 的有限差分近似. 然后如下定义差分格式: 用网格上的近似值代替真解并且让关系式 (9.6) 的两边相等,

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = f_{ij} \tag{9.7}$$

$$2 \leqslant i \leqslant n_x, 2 \leqslant j \leqslant n_y$$

用离散的狄利克雷边界条件

$$u_{ij} = g_{ij} \quad i = 1 \text{ 或 } n_x + 1, \text{ 或者 } j = 1 \text{ 或 } n_y + 1 \tag{9.8}$$

补充这些方程.

所得到的方法称为五点格式, 因为 (9.7) 中的每个差分方程是一个内部网格点和其四个相邻网格点上函数值的关系, 见图 9-2. 因为边界网格点上的近似解由 (9.8) 给出, 所以未知量是内部网格点上的近似解值 $u_{ij}, 2 \leqslant i \leqslant n_x, 2 \leqslant j \leqslant n_y$, 它们共有 $(n_x-1)(n_y-1)$ 个. 未知量个数与 (9.7) 中的方程个数相匹配. 实际上可以证明, 格式 (9.7~9.8) 有唯一解.

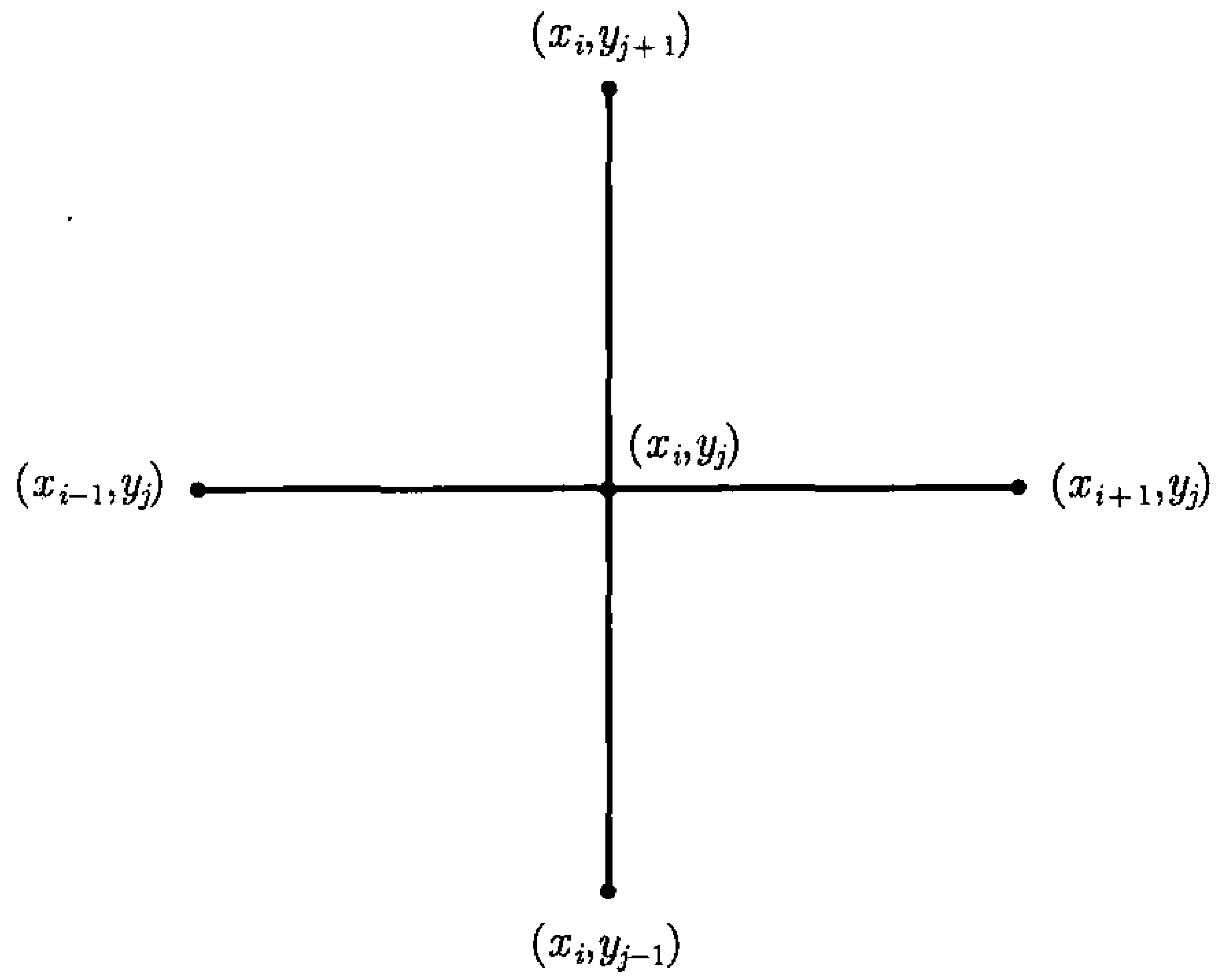


图 9-2 内部网格点及其四个相邻的网格点

规定格式 (9.7) 的截断误差为

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h_x^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{h_y^2} - f_{ij}$$

若解 u 有 4 阶连续的导数, 则截断误差为 $O(h_x^2 + h_y^2)$. 用泰勒定理可验证这个结论. 可以证明解的精度为 2 阶:

$$\max_{\substack{1 \leqslant i \leqslant n_x+1 \\ 1 \leqslant j \leqslant n_y+1}} |u(x_i, y_j) - u_{ij}| = O(h_x^2 + h_y^2) \tag{9.9}$$

注意, 讨论常微分方程初值问题的数值解时, 截断误差是指在网格点上格式解的局部误差. 分析求解边值问题的数值方法时, 规定截断误差是差分方程和微分方程之间的偏差. 对于狄利克雷边界条件, 求解 (9.5) 的方法 (9.7) 的收敛阶与截断误差的阶相同.

下面集中讨论特殊情况 $n_x = n_y = n$, $h_x = h_y = h$. 于是差分方程 (9.7) 可写成

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2 f_{ij}, \quad 2 \leqslant i,j \leqslant n \tag{9.10}$$

离散方程 (9.10) 的形式自然地使人想到求解的迭代方法. 把格式写成

$$\begin{aligned} &u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - \frac{h^2}{4} f_{ij}, \quad 2 \leqslant i,j \leqslant n \\ &u_{i,j} = g_{i,j}, \quad i = 1 \,\text{or}\, n+1, 2 \leqslant j \leqslant n, \text{ or } j = 1 \,\text{or}\, n+1, 2 \leqslant i \leqslant n \end{aligned} \tag{9.11}$$

接着实施 6.6 节的高斯–赛德尔迭代法如下：给定初始猜测 $\{u_{i,j}^{(1)}\}_{2\leqslant i,j\leqslant n}$, 对 $k = 1, 2, \cdots$, 通过

for $i = 2, \cdots, n$

  for $j = 2, \cdots, n$

$$u_{i,j}^{(k+1)} = \frac{1}{4}\left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)}\right) - \frac{h^2}{4} f_{ij}$$

  end

end

递归地确定 $\{u_{i,j}^{(k+1)}\}_{2\leqslant i,j\leqslant n}$.

当 i 或 j 是 1 或 $n+1$ 时, 使用边值条件: $u_{i,j}^{(k+1)} = g_{i,j}$. 在这个实施过程中, 对第 2 个指标做内部迭代, 而对第 1 个指标做外部迭代. 可以交换内部迭代和外部迭代的次序得到另一种高斯–赛德尔实施方法.

MATLAB 程序 下面是单位正方形 $[0,1]^2 \equiv [0,1] \times [0,1]$ 上求解泊松方程的狄利克雷边值问题的一个 MATLAB 程序:

```
function U = Poisson(f,g,n,tol,max_it)
%
% function U = Poisson(f,g,n,tol,max_it)
%
% The five-point scheme for solving the Dirichlet BVP of the
% Poisson equation on the unit square.
% Input
%      f: the right-hand-side function
%      g: the Dirichlet boundary value function
%      n: the number of subintervals of [0,1]
%      tol: relative error tolerance of the iterative solution;
%           default value: 10^(-5)
%      max_it: maximal number of iterations allowed;
%           default value: 10,000
```

```
% Output
%       U: the solution u_{ij}, i,j=1,...,n+1
%
% To use the program, the user must supply two m-files, say,
% 'f.m' and 'g.m', to define the right-hand-side function f of
% the differential equation and the boundary value function g.
% The user should also choose a positive integer n for the
% number of subintervals of [0,1].
% A sample call would be
%      U = Poisson('f','g',n,1e-8,1000)
% with 10^(-8) as the tolerance for the relative errors of the
% iterative solution, and a maximal number of 1,000 iterations
% is allowed for solving the finite difference system.
% It is also possible to use
%     U = Poisson('f','g',n,1e-8)
% then the maximal number of iterations is the default
% value 10^4. If the default values 10^(-5) and 10^4 are
% to be used for the iteration relative error tolerance and
% maximal number of iterations, then one can simply use
%     U = Poisson('f','g',n)
% if max_it, or max_it and tol, are not provided, use the
% default values
if nargin < 5
  max_it = 10000
end
if nargin < 4
  tol = 1e-5
end
% compute some parameters
n1 = n+1;
h = 1/n;
toln = (h^2)*tol;
Fr = zeros(n,n);
Fr(2:n,2:n) = h2*feval(f,(1:n-1)*h,(1:n-1)*h);
% initialization
U = zeros(n1,n1);
% specify boundary conditions
U(1,1:n1) = feval(g,0,(0:n)*h);
U(n1,1:n1) = feval(g,1,(0:n)*h);
U(1:n1,1) = feval(g,(0:n)*h,0);
U(1:n1,n1) = feval(g,(0:n)*h,1);
% iteration
rel_err = 1;
itnum = 0;
while ((rel_err>toln) & (itnum<=max_it))
err = 0;
```

```
umax = 0;
  for j = 2:n
    for i = 2:n
      temp = (U(i+1,j)+U(i-1,j)+U(i,j+1)+U(i,j-1))/4-Fr(i,j);
      dif = abs(temp-U(i,j));
      if (err <= dif)
        err = dif;
      end
      U(i,j) = temp;
      temp = abs(temp);
      if(umax <= temp)
        umax = temp;
      end
    end
  end
itnum = itnum+1;
rel_err = err/umax;
end
% plot the numerical solution
X=(0:h:n*h)';
Y=X;
surf(X,Y,U')
xlabel('x-axis')
ylabel('y-axis')
zlabel('the numerical solution')
title('Plot of the numerical solution')
```

在程序中, 使用 MATLAB 指令`surf`画数值解的 3 维图形. 注意, 关于语句`surf` $(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z})$, 曲面轨迹的顶点是三元组 $(\boldsymbol{X}(j),\boldsymbol{Y}(i),\boldsymbol{Z}(i,j))$, 即第一个向量 $\boldsymbol{X}$ 对应于矩阵 $\boldsymbol{Z}$ 的列, 而第二个向量 $\boldsymbol{Y}$ 对应于矩阵 $\boldsymbol{Z}$ 的行. 上面的程序中, (x_i,y_j) 上的数值解用 $u(i,j)$ 表示. 这就是为什么在语句`surf(X,Y,U')` 中使用转置 `U'`. 同样的注记应用于其他 MATLAB 三维指令: `mesh`, `waterfall`, `surfl`以及它们的彩色副本.

例 9.1.1 考虑边值问题

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=-2\pi^2\sin(\pi x)\sin(\pi y), \quad 0<x, \quad y<1 \\ u(x,y)=0, \quad x=0,1\text{或}y=0,1 \end{cases}$$

真解是

$$u(x,y)=\sin(\pi x)\sin(\pi y)$$

它在图 9-3 中展示.

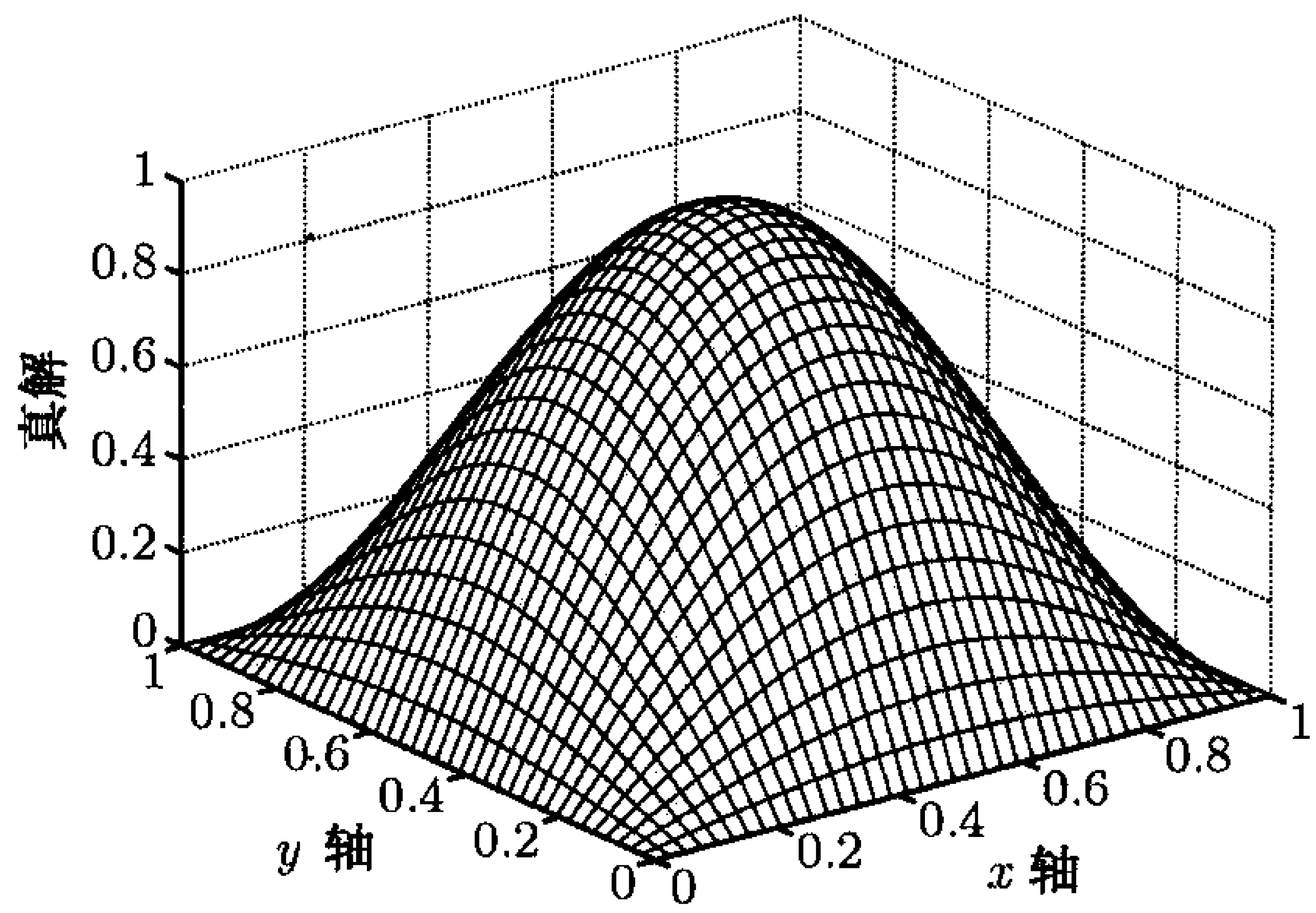


图 9-3 例 9.1.1 的真解

用五点格式求解上述问题. 此例中, 我们知道真解, 因此可以计算数值解的误差. 表 9-1 对几个 n 列出极大误差 $\max_{1\leqslant i,j\leqslant n+1}|u(x_i,y_i)-u_{ij}|$. ■

表 9-1 例 9.1.1 的极大数值解误差

n	极大误差	比率
4	5.3029E−2	
8	1.2951E−2	4.09
16	3.2190E−3	4.02
32	8.0347E−4	4.01

图 9-4 显示 $n=16$ 时的数值结果. 当用下列程序代替程序Poisson的绘图部分时生成图 9-4.

```
% Plot the numerical solution
X = (0:h:n*h)'; Y = X;
subplot(1,2,1)
surf(X,Y,U')
xlabel('x-axis')
ylabel('y-axis')
zlabel('The numerical solution')
s1 = sprintf('h=6.4f',h)
title(s1)
hold on
%
% Plot the numerical error
Err = sin(pi*X)*sin(pi*Y')-U;
subplot(1,2,2)
surf(X,Y,Err')
```

```
xlabel('x-axis')
ylabel('y-axis')
zlabel('The error')
title(s1)
```

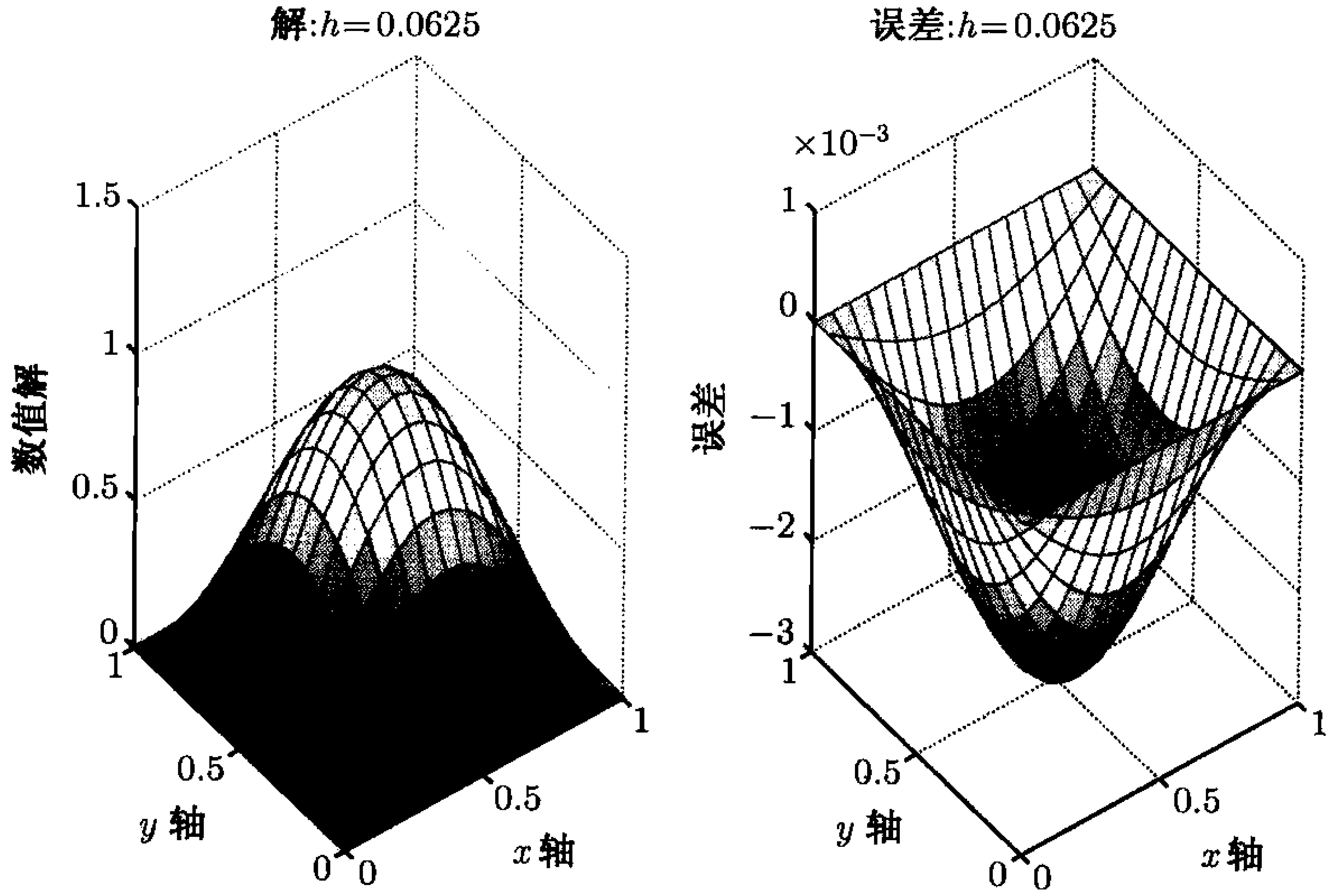


图 9-4 $n = 16$ 的数值结果

从表 9-1 可以看出, 当 n 的值加倍时, 即网格尺寸 h 折半时, 最大误差大约减少到原来的 1/4. 这与理论上的误差界 (9.9) 一致. 为更精细地察看误差性态, 我们在表 9-2 中给出六个结点上解的误差和对应的比率. 这些比率都接近于 4. 注意到边值问题具有各种对称性质. 例如, 它关于问题区域的对角线和中线是对称的, 对角线和中线分别用方程 $y = x$, $x + y = 1$, $x = 1/2$ 和 $y = 1/2$ 定义. 这就说明为什么几个点上的误差是相等的. 为了检查给出的代码是否有错, 一项容易的工作是查看代码是否保持求解问题的对称性.

表 9.2 例 9.1.1 的数值解误差

(x, y)	$n=4$	$n=8$	比率	$n=16$	比率
(1/4, 1/4)	−2.65E−2	−6.48E−3	4.09	−1.61E−3	4.02
(1/2, 1/4)	−3.75E−2	−9.16E−3	4.09	−2.28E−3	4.02
(3/4, 1/4)	−2.65E−2	−6.48E−3	4.09	−1.61E−3	4.02
(1/4, 1/2)	−3.75E−2	−9.16E−3	4.09	−2.28E−3	4.02
(1/2, 1/2)	−5.30E−2	−1.30E−2	4.09	−3.22E−3	4.02
(1/4, 3/4)	−2.65E−2	−6.48E−3	4.09	−1.61E−3	4.02

我们可以改进理论上的误差界 (9.9). 在解的某些光滑性假设之下, 可以证明五

点格式的数值解, 对任何网格点 (x_i, y_j) 有一个渐近的误差展开

$$u(x_i, y_j) - u_{ij} = h^2 D(x_i, y_j) + O(h^4) \tag{9.12}$$

这里 $D(x, y)$ 表示由数据中包含 u 的一个边值问题确定的某个函数. 这就使执行理查森外推成为可能. 用 $u_h(x, y)$ 表示具有网格尺寸 h 的网格点 (x, y) 上的数值解, 改写 (9.12) 为

$$u(x_i, y_j) - u_h(x_i, y_j) = h^2 D(x_i, y_j) + O(h^4) \tag{9.13}$$

若 (x, y) 是对应于网格尺寸 $2h$ 的网格点, 则它也是具有网格尺寸 h 的网格点. 由 (9.13) 有

$$\begin{aligned} u(x, y) - u_h(x, y) &= h^2 D(x, y) + O(h^4) \\ u(x, y) - u_{2h}(x, y) &= (2h)^2 D(x, y) + O(h^4) \end{aligned}$$

从这两个关系式消去 $D(x, y)$ 项, 得到

$$u(x, y) - \tilde{u}_h(x, y) = O(h^4)$$

其中 $\tilde{u}_h$ 是由下式定义的外推解

$$\tilde{u}_h(x, y) = \frac{4u_h(x, y) - u_{2h}(x, y)}{3}$$

我们在表 9.3 中给出外推解的误差. 注意到数值解精度的有效改进. 因为外推解的误差相当小, 所以必须更精确地求解有限差分方程组. 可在运行程序`Poisson`中使用小的误差容限来完成这项工作. 表 9-3 中给出的结果对应于 $\text{tol} = 10^{-9}$.

表 9-3 例 9.1.1 的外推解误差

(x, y)	$n = 8$	$n = 16$	比率
(1/4, 1/4)	2.04E−4	1.25E−5	16.35
(1/3, 1/4)	2.89E−4	1.77E−5	16.35
(3/4, 1/4)	2.04E−4	1.25E−5	16.35
(1/4, 1/2)	2.89E−4	1.77E−5	16.35
(1/2, 1/2)	4.09E−4	2.50E−5	16.35
(1/4, 3/4)	2.04E−4	1.25E−5	16.35

表 9-2 和表 9-3 给出的数值结果是通过下列 MATLAB 代码得到的, 其中`fP1`定义右端函数, `gP1`定义边值函数.

```
% Define x and y coordinates of sampling points
xx = linspace(0.25,0.75,3)';
yy = xx;
% True solution at 9 points
tr = sin(pi*xx)*sin(pi*yy)'
% True solution values at 6 points (1/4,1/4),(1/2,1/4),...
Tr = [tr(1,1);tr(2,1);tr(3,1);tr(1,2);tr(2,2);tr(1,3)];
% Compute solution errors
N = 2.^(2:4)
err = zeros(6,3);
for i = 1:3
   n = N(i);
   n4 = n/4;
   zz = Poisson('fP1','gP1',n,1e-9);
   zv = zz(n4+1:n4:n,n4+1:n4:n);
   Zv = [zv(1,1);zv(2,1);zv(3,1);zv(1,2);zv(2,2);zv(1,3)];
   err(:,i) = Tr-Zv;
end
% Compute solution error ratios
R1 = zeros(6,3);
R1(:,1) = err(:,1)./err(:,2);
R1(:,2) = err(:,2)./err(:,3);
% Extrapolation
exte = zeros(6,2);
for i = 1:2
   exte(:,i) = (4.*err(:,i+1)-err(:,i))/3;
end
% Compute ratios of the extrapolation errors
R2 = zeros(6,1);
R2(:)  = exte(:,1)./exte(:,2);
% Print results
for i = 1:6
   fprintf(' 10.2e%10.2e%6.2f%10.2e%6.2f\n',...
   err(i,1),err(i,2),R1(i,1),err(i,3),R1(i,2))
end
for i = 1:6
   fprintf(' 10.2e%10.2e%6.2f\n',...
             exte(i,1),exte(i,2),R2(i))
end
```

习题

1. 回顾双曲函数的定义

$$\sinh(x) = \frac{e^x - e^{-x}}{2}, \quad \cosh(x) = \frac{e^x + e^{-x}}{2}$$

证明函数

$$u(x,y)=\frac{1}{\sinh(\pi)}\left[\sin(x)\sinh(\pi-y)+\sin(y)\sinh(\pi-x)\right]$$

是边值问题

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=0, & 0<x,y<\pi\\ u(x,0)=\sin(x),\quad u(x,\pi)=0, & 0\leqslant x\leqslant\pi\\ u(0,y)=\sin(y),\quad u(\pi,y)=0, & 0\leqslant y\leqslant\pi\end{cases}$$

的解.

2. 若 $f(z)$ 是一个可微的复变函数, 其中 $z=x+iy$, 则函数 $f(x+iy)$ 的实部 $u(x,y)$ 和虚部 $v(x,y)$ 满足柯西–黎曼 (Cauchy-Riemann) 方程

$$u_x=v_y,\quad u_y=-v_x$$

而且, u 和 v 是无限次可微的. 证明 u 和 v 都满足拉普拉斯方程.

3. 证明 $u(x,y)=(x-x^4)(y-y^2)$ 是边值问题

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=2x(x^3-6xy+6xy^2-1), & 0<x,y<1\\ u(x,y)=0, & x=0\text{或}1,\text{或者}\quad y=0\text{或}1\end{cases}$$

的解. 利用 MATLAB 程序Poisson解此问题. 类似于例 9.1.1 给出数值结果.

4. 修改程序Poisson使得它在单位正方形的四条边上取不同表达式的边值函数, 即编写下列形式的程序

```
function U = Poisson1(f,gr,gt,gl,gb,n,tol,max_it)
```

其中, gr, gt, gl和gb是在边界的右面部分 $(x=1)$、顶部 $(y=1)$、左面部分 $(x=0)$ 和底部 $(y=0)$ 上边值条件的函数名称. 利用代码求解下列问题:

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=-2(x^2+y^2), & 0<x,y<1\\ u(x,0)=1-x^2,\quad u(x,1)=2(1-x^2), & 0\leqslant x\leqslant 1\\ u(0,y)=1+y^2,\quad u(1,y)=0, & 0\leqslant y\leqslant 1\end{cases}$$

真解是 $u(x,y)=(1-x^2)(1+y^2)$. 对这个特殊问题, 五点格式在网格点上能再现真解. 如表 9-1 中那样, 通过对几个相对误差容限的值(max_it=10000): $10^{-5},10^{-6},10^{-7},10^{-8}$ 计算最大的数值解误差来确认这个性质.

5. 用习题 4 中的程序求解边值问题

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=0, & 0<x,y<1\\ u(x,0)=1,\quad u(x,1)=2, & 0<x<1\\ u(0,y)=1+y^2,\quad u(1,y)=1+y, & 0<y<1\end{cases}$$

取 $n_x=n_y=5,10,20$. 画出对应的数值解图形. 作一个在点 $(i/5,i/5)(1\leqslant i\leqslant 5)$ 上的数值表.

6. 如前面的习题一样修改程序Poisson, 使得它在单位正方形边界的四条边上取不同表达式的边值函数, 而且, 可以求解的微分方程具有更一般的形式

$$a_1\frac{\partial^2 u}{\partial x^2}+a_2\frac{\partial^2 u}{\partial y^2}=f(x,y)$$

这里 a_1 和 a_2 是正常数. 利用新的程序求解下列问题:

$$\begin{cases}\pi^2\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=0, & 0<x,y<1\\ u(x,0)=u(x,1)=0, & 0\leqslant x\leqslant 1\\ u(0,y)=\sin\pi y,\quad u(1,y)=\mathrm{e}\sin\pi y, & 0\leqslant y\leqslant 1\end{cases}$$

真解是 $u(x,y)=\mathrm{e}^x\sin\pi y$. 类似于例 9.1.1 的样式给出数值结果.

7. 修改程序Poisson使得它能够在任意边长 $a>0$ 的正方形 $[0,a]^2$ 上求解泊松方程的狄利克雷边值问题. 利用新程序求解下列问题:

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=(6y-y^3)\sin x, & 0<x,y<\pi\\ u(x,0)=0,\quad u(x,\pi)=\pi^3\sin x, & 0\leqslant x\leqslant \pi\\ u(0,y)=u(\pi,y)=0, & 0\leqslant y\leqslant \pi\end{cases}$$

真解是 $u(x,y)=y^3\sin x$. 按类似于例 9.1.1 的样式给出数值结果.

8. 偏微分方程

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+a(x,y)u=f(x,y)$$

称为亥姆霍兹 (Helmholtz) 方程. 当系数函数 $a(x,y)=0$ 时, 它简化为泊松方程. 考虑非正的常系数 a 的特殊情况及相应的边值问题

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}+au=f(x,y), & 0<x,y<1\\ u(x,y)=g(x,y), & x=0\text{或}1,\quad \text{或者 } y=0\text{或}1\end{cases}$$

导出一个类似于 (9.11) 的差分格式, 求解这个边值问题. 按 MATLAB 实施这个格式并利用它关于 $a=-2$, $g(x,y)=0$ 和

$$f(x,y)=xy[(x^2-7)(1-y^2)+(1-x^2)(y^2-7)]$$

求解边值问题. 真解是

$$u(x,y)=(x-x^3)(y-y^3)$$

类似于例 9.1.1 的格式给出数值结果.

9. 对于给定的函数 f,g_l,g_r,g_b,g_t, 考虑边值问题

$$\begin{aligned}&\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y), \quad 0<x,y<1\\ &u_x(0,y)=g_l(y),\quad u(1,y)=g_r(y),\quad 0\leqslant y\leqslant 1\\ &u(x,0)=g_b(x),\quad u(x,1)=g_t(x),\quad 0\leqslant x\leqslant 1\end{aligned}$$

指出如何利用公式 (8.143) 的思想去离散导数边界条件 $u_x(0,y)=g_l(y)$.

9.2　一维热传导方程

本节研究的模型抛物型方程是

$$\frac{\partial u}{\partial t}=a\frac{\partial^2 u}{\partial x^2}+f \tag{9.14}$$

这里 t 是时间变量, 而 $x\in(0,L)$ 是空间变量. 许多物理现象 (包括热传导问题) 的模型都可使用这样的方程. 为得到一个明确定义 (well-defined) 的问题, 必须用初始条件

$$u(x,0)=u_0(x),\quad x\in[0,L] \tag{9.15}$$

和边界条件

$$u(0,t)=g_1(t)\quad u(L,t)=g_2(t) \tag{9.16}$$

补充微分方程. 我们对 $t\leqslant T$ 求解初边值问题. 微分方程 (9.14)、条件 (9.15) 和条件 (9.16) 一起构成一个初边值问题. 见图 9-5.

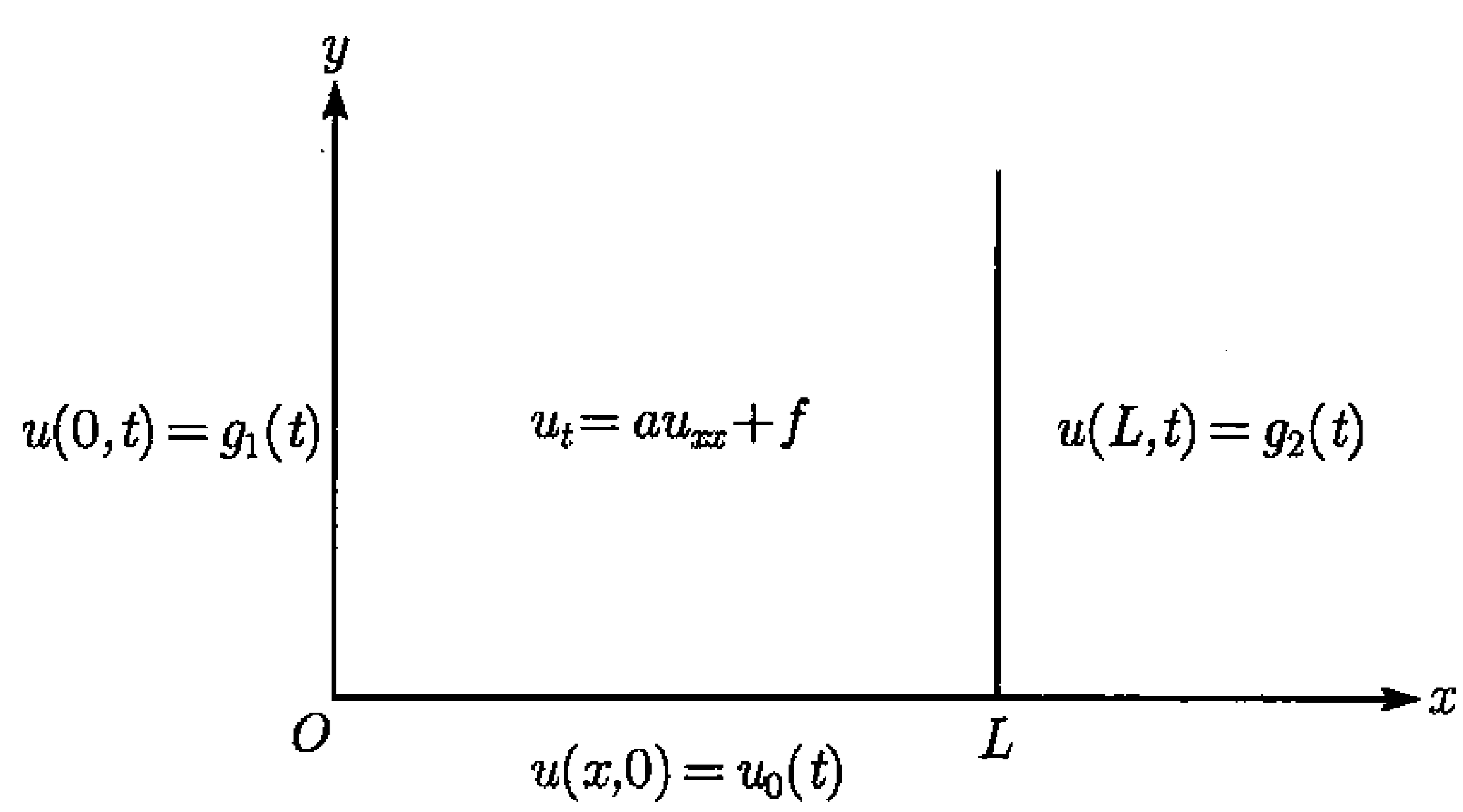


图 9-5　抛物型方程的初边值问题

求解初边值问题 (9.14~9.16) 的数值方法的设计可以采取两种方法. 第一种方法, 推导数值方法有两个步骤: 第一步离散空间导数, 导致一个半离散系统, 这是一个关于时间变量的常微分方程组; 第二步利用 ODE 解算器求解所得到的常微分方程组 (ODE). 第二种方法, 同时离散两个有关的变量 x 和 t, 从而得到立即可以使用和求解的全离散格式.

9.2.1　半离散化

首先讨论半离散化方法. 为此, 把空间 $[0,L]$ 分割 n_x 等分. 网格大小用 $h_x=L/n_x$ 表示, 网格点为 $x_i=(i-1)h_x, 1\leqslant i\leqslant n_x+1$. 设 $u_i(t)$ 是 $u(x_i,t)$ 的近似值, $1\leqslant i\leqslant n_x+1$. 对 $2\leqslant i\leqslant n_x$, 考虑微分方程 (9.14) 并利用三点中心差分公式

$$\frac{\partial^2 u}{\partial x^2}(x_i,t)\approx\frac{u(x_{i+1},t)-2u(x_i,t)+u(x_{i-1},t)}{h_x^2}$$

去作逼近. 结果得到

$$\dot{u}_i(t) = a\frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{h_x^2} + f_i(t), \quad 2 \leqslant i \leqslant n_x$$

其中 $f_i(t) \equiv f(x_i, t)$, 量上面的点表示这个量的时间导数. 这些方程再补充从边值条件 (9.16) 得到的关系式

$$u_1(t) = g_1(t), \quad u_{n_x+1}(t) = g_2(t)$$

以及由 (9.15) 得到的初值

$$u_i(0) = u_0(x_i), \quad 1 \leqslant i \leqslant n_x + 1$$

按矩形/向量记号, ODE 方程组可写成

$$\begin{aligned} \dot{\boldsymbol{u}}(t) &= \boldsymbol{A}\boldsymbol{u}(t) + \boldsymbol{F}(t) \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0 \end{aligned} \tag{9.17}$$

其中 $\boldsymbol{u}(t) = [u_2(t), \cdots, u_{n_x}(t)]^{\mathrm{T}}$ 是未知向量,

$$\boldsymbol{A} = \frac{a}{h_x^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}_{(n_x-1)\times(n_x-1)}$$

$$\boldsymbol{F}(t) = \left[f_2(t) + \frac{a}{h_x^2}g_1(t), f_3(t), \cdots, f_{n_x-1}(t), f_{n_x}(t) + \frac{a}{h_x^2}g_2(t)\right]^{\mathrm{T}}$$

$$\boldsymbol{u}_0 = [u_0(x_2), \cdots, u_0(x_{n_x})]^{\mathrm{T}}$$

一般地, 实际计算中使用的典型值 n_x 会使得 ODE 方程组 (9.17) 具有刚性. 故应该用对刚性方程有效的 ODE 解算器求解这个方程. 这种方法的主要优点是可以利用最新技术水平的 ODE 解算器.

9.2.2 显式全离散化

现在转到全离散格式的讨论. 除了空间区间 $[0, L]$ 的划分外, 还需要对时间变量选择步长. 对某个正整数 n_t, 记 $h_t = T/n_t$, 而网格点为 $t_k = (k-1)h_t$, $1 \leqslant k \leqslant n_t+1$. 用 u_i^k 表示 $u(x_i, t_k)$ 的有限差分近似值. 可在 (x_i, t_k) 处离散微分方程, $2 \leqslant i \leqslant n_x$, $2 \leqslant k \leqslant n_t$. 对二阶的空间导数仍利用三点中心差分逼近

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_k) \approx \frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{h_x^2}$$

逼近时间导数项 $\frac{\partial u}{\partial t}(x_i,t_k)$ 可有几种方法, 如用

$$\frac{\partial u}{\partial t}(x_i,t_k) \approx \frac{u_i^{k+1}-u_i^k}{h_t}$$

就得到向前欧拉公式

$$\frac{u_i^{k+1}-u_i^k}{h_t} = a\frac{u_{i+1}^k-2u_i^k+u_{i-1}^k}{h_x^2}+f_i^k \tag{9.18}$$

定义 (9.18) 的截断误差为

$$T_i^k \equiv \frac{u(x_i,t_{k+1})-u(x_i,t_k)}{h_t} - a\frac{u(x_{i+1},t_k)-2u(x_i,t_k)+u(x_{i-1},t_k)}{h_x^2} - f(x_i,t_k)$$

因为 $(u(x_i,t_{k+1})-u(x_i,t_k))/h_t$ 是导数 $u_t(x_i,t_k)$ 的 $O(h_t)$ 近似, 而 $(u(x_{i+1},t_k)-2u(x_i,t_k)+u(x_{i-1},t_k))/h_x^2$ 是 $u_{xx}(x_i,t_k)$ 的 $O(h_x^2)$ 近似, 所以

$$\begin{aligned} T_i^k &= u_t(x_i,t_k)+O(h_t)-au_{xx}(x_i,t_k)+O(h_x^2)-f(x_i,t_k) \\ &= O(h_t+h_x^2) \end{aligned}$$

定义比值

$$\gamma = \frac{ah_t}{h_x^2}$$

结合初值条件和边值条件, 方法的完整描述是

$$u_i^1 = u_0(x_i), \quad 1 \leqslant i \leqslant n_x+1 \tag{9.19}$$

并且对 $k \geqslant 1$,

$$\begin{cases} u_i^{k+1} = \gamma u_{i-1}^k + (1-2\gamma)u_i^k + \gamma u_{i+1}^k + h_t f_i^k, \quad 2 \leqslant i \leqslant n_x \\ u_1^{k+1} = g_1(t_{k+1}), \quad u_{n_x+1}^{k+1} = g_2(t_{k+1}) \end{cases} \tag{9.20}$$

在这个方法中, 一旦知道 $t=t_k$ 层的近似解, 就可以直接计算在下一个时间层 $t=t_{k+1}$ 上的解. 故此方法称为*显式方法*.

显式方法的主要优点是不必求解线性方程组, 因为在时间层 $t=t_{k+1}$ 上的近似解可直接从前面的时间层上的近似解算出. 然而, 一个有用的数值方法必须是稳定的, 即随着时间向前推进, 误差传播应该可控. 假定 $t=t_k$ 层上的计算解是 $\tilde{u}_i^k = u_i^k+\varepsilon_i^k$, $\left|\varepsilon_i^k\right| \leqslant \varepsilon^k$, $1 \leqslant i \leqslant n_x+1$. 则容易从 (9.20) 得到 $t=t_{k+1}$ 层上的计算解 (假定没有引入进一步的误差) 为

$$\tilde{u}_i^{k+1} \equiv u_i^{k+1}+\varepsilon_i^{k+1} = \gamma\tilde{u}_{i+1}^k + (1-2\gamma)\tilde{u}_i^k + \gamma\tilde{u}_{i-1}^k + h_t f_i^k$$

因此

$$\varepsilon_i^{k+1} = \gamma\varepsilon_{i+1}^k + (1-2\gamma)\varepsilon_i^k + \gamma\varepsilon_{i-1}^k, \quad 2 \leqslant i \leqslant n_x$$

是对应于解 u_i^{k+1} 的误差, $2 \leqslant i \leqslant n_x$. 注意到, 若

$$\gamma \equiv \frac{ah_t}{h_x^2} \leqslant \frac{1}{2} \tag{9.21}$$

则

$$\left|\varepsilon_i^{k+1}\right| \leqslant \gamma\left|\varepsilon_{i+1}^k\right| + (1-2\gamma)\left|\varepsilon_i^k\right| + \gamma\left|\varepsilon_{i-1}^k\right| \leqslant \varepsilon^k, \quad 2 \leqslant i \leqslant n_x$$

即随时间向前推进, 误差不被放大, 因此数值方法是稳定的. 可以证明 (9.21) 实际上是格式 (9.19~9.20) 稳定的充要条件, 因而它是有用的.

假定稳定性条件 (9.21) 成立. 可以证明当真解 u 充分光滑时, 下列误差界成立:

$$\max_{\substack{1\leqslant i\leqslant n_x+1\\1\leqslant k\leqslant n_t+1}} \left|u(x_i,t_k) - u_i^k\right| = O(h_t + h_x^2)$$

即格式具有 h_x 的 2 阶精度和 h_t 的 1 阶精度. 注意收敛阶与截断误差的阶相同.

MATLAB 程序 下面是显式方法的一个 MATLAB 程序.

```
function U = Heat1(a,f,u0,g1,g2,L,T,nx,nt)
%
% function U = Heat1(a,f,u0,g1,g2,L,T,nx,nt)
%
% The forward difference scheme for solving the initial boundary
% value problem of the heat equation u_t = a u_{xx} + f for x
% in [0,L], and t in [0,T].
% Input
%      a: the coefficient of the u_{xx} term
%      f=f(x,t): the right-hand-side function
%      u0=u0(x): the initial value function
%      g1=g1(t): the boundary value function at x=0
%      g2=g2(t): the boundary value function at x=L
%      L:  right end-point of the spatial interval
%      T:  right end-point of the time interval
%      nx: the number of subintervals of [0,L]
%      nt: the number of subintervals of [0,T]
% Output
%       U: the solution (u_{k,i}), k the time grid point index,
%          i the space grid point index
% compute some parameters
```

```
nx1 = nx+1;
hx = L/nx;
nt1 = nt+1;
ht = T/nt;
r = a*ht/(hx*hx)
r1 = 1-2*r;
% stability test
if r > 0.5
  disp('The ratio is too big; the scheme is unstable!')
end
% generate grid point vectors
xvec = hx*(0:nx);
tvec = ht*(0:nt);
% initialization
U = zeros(nt1,nx1);
U(1,:) = feval(u0,xvec(:)');
% time advancing
U(2:nt1,1) = feval(g1,(1:nt)*ht);
U(2:nt1,nx1) = feval(g2,(1:nt)*ht);
for k = 1:nt
  U(k+1,2:nx) = r*(U(k,1:nx-1)+U(k,3:nx+1))+r1*U(k,2:nx) ...
                +ht*feval(f,xvec(2:nx),tvec(k));
end
% plot the numerical solution
surf(xvec,tvec,U)
xlabel('x-axis')
ylabel('t-axis')
zlabel('the numerical solution')
s1 = sprintf('h_t=%6.4f  h_x=%6.4f',ht,hx)
title(s1)
```

例 9.2.1 对 $t \leqslant 0.2$ 求解下列初边值问题:

$$\begin{cases} u_t = u_{xx}, & x \in (0,1), \quad t > 0 \\ u(x,0) = \sin(\pi x), & x \in [0,1] \\ u(0,t) = u(1,t) = 0, & t > 0 \end{cases}$$

真解是 $u(x,t) = \mathrm{e}^{-\pi^2 t}\sin(\pi x)$, 如图 9-6 所示.

此例中, $a = 1$, $L = 1$, $T = 0.2$. 于是 $h_t = 0.2/n_t$, $h_x = 1/n_x$. 故稳定性条件 (9.21) 为

$$\gamma = \frac{0.2 n_x^2}{n_t} \leqslant \frac{1}{2} \tag{9.22}$$

选择 $n_x = 3, 6, 12$ 以及相应的 $n_t = 6, 24, 96$, 故 $\gamma = 0.3$, 稳定性条件 (9.22) 满足. 表 9-4 中, 对 $t_k = 0.2$ 列出极大误差 $\max_{1 \leqslant i \leqslant n_x+1} \left|u(x_i, t_k) - u_i^h\right|$.

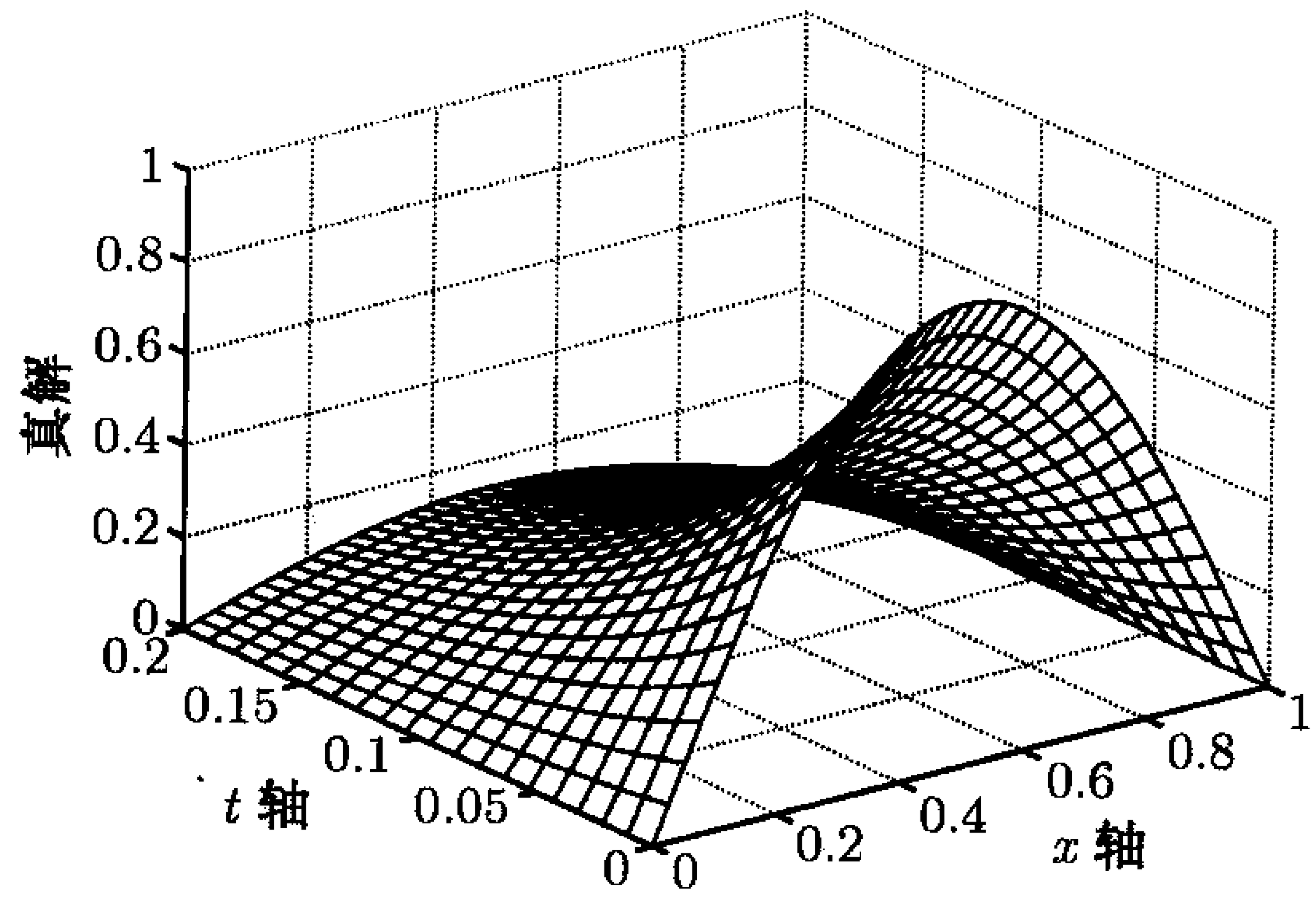


图 9-6 真解

表 9-4 t=0.2 处解的极大误差

n_x	n_t	极大误差
3	6	1.8414E−2
6	24	5.0829E−3
12	96	1.2573E−3

$n_x = 12$ 和 $n_t = 96$ 的数值解和对应的误差如图 9-7 所示. 这个结果是用下列程序替代程序Heat1的最后部分完成的.

```
% Plot the numerical solution
subplot(1,2,1)
surf(xvec,tvec,U)
xlabel('x-axis')
ylabel('t-axis')
zlabel('the numerical solution')
s1 = sprintf('h_t=%6.4f h_x=%6.4f',ht,hx)
title(s1)
hold on
% Plot the solution error
Err = exp(-pi*pi*tvec)'*sin(pi*xvec)-U;
subplot(1,2,2)
surf(xvec,tvec,Err)
xlabel('x-axis')
ylabel('t-axis')
zlabel('the error')
title(s1)
```

■

例 9.2.2 为观察稳定性条件 (9.21) 扮演的角色, 我们继续讨论例 9.2.1 中初边值问题的数值解. 首先取 $n_x = 10$ 和 $n_t = 10$ 求解问题. 计算结果如图 9-8 所示. 此时, $\gamma = 2$, 因而格式不稳定. 然而, 因为只应用了 10 次 ($n_t = 10$) 差分格式, 所以舍入误差的累积不明显.

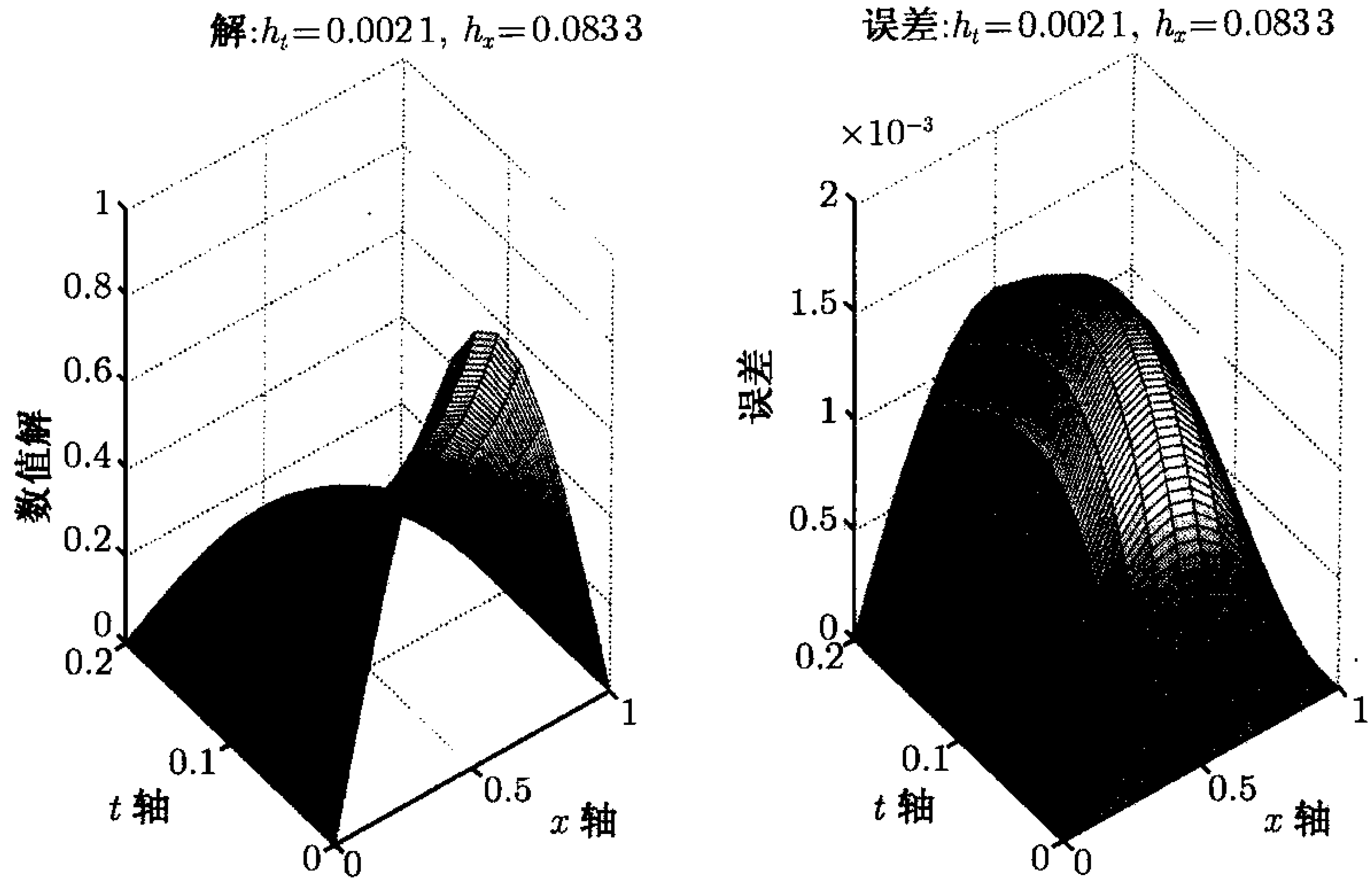


图 9-7　显式方法的数值结果：$n_x = 12$, $n_t = 96$

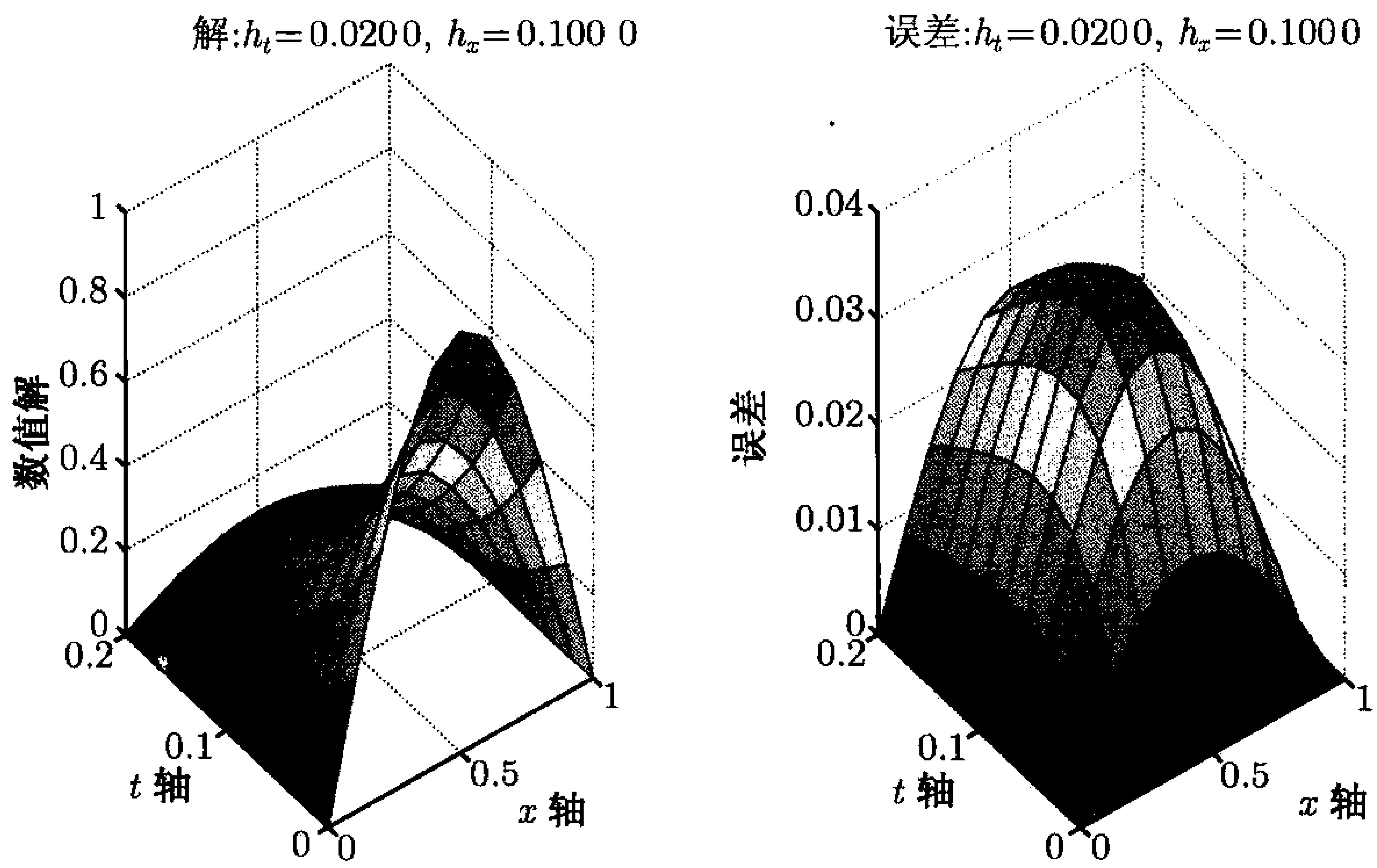


图 9-8　显式方法的数值结果：$n_x = 20$, $n_t = 10$

接着取 $n_x = 20$ 和 $n_t = 20$. 此时 $\gamma = 4$. 计算结果如图 9-9 所示. 观察到, t 接近于它的上界 0.2 时, 舍入误差累积的影响变得相当明显.

此例表明使用显式格式 (9.19~9.20) 时保持稳定性条件 (9.21) 的重要性.　■

9.2.3　隐式全离散化

我们已经看到, 对显式全离散格式, 应该选择 h_t 和 h_x 以满足稳定性条件 (9.21). 条件 (9.21) 对 h_t 关于 h_x 的相对大小强加了一个限制. 假如对应于所选

的 h_t 和 h_x 的数值解不够精确, 则很自然地要去尝试用更小的 h_t 和 h_x. 假设我们使用 $\bar{h}_x = h_x/2$. 则为了保持相同的比值 γ, 需选择 $\bar{h}_t$ 使得

$$\frac{a\bar{h}_t}{\bar{h}_x^2} = \frac{ah_t}{h_x^2}$$

即

$$\bar{h}_t = \frac{1}{4}h_t$$

换句话说, 每当 h_x 折半时, h_t 必须缩小到原来的 1/4. 这可能导致过分小的时间步长.

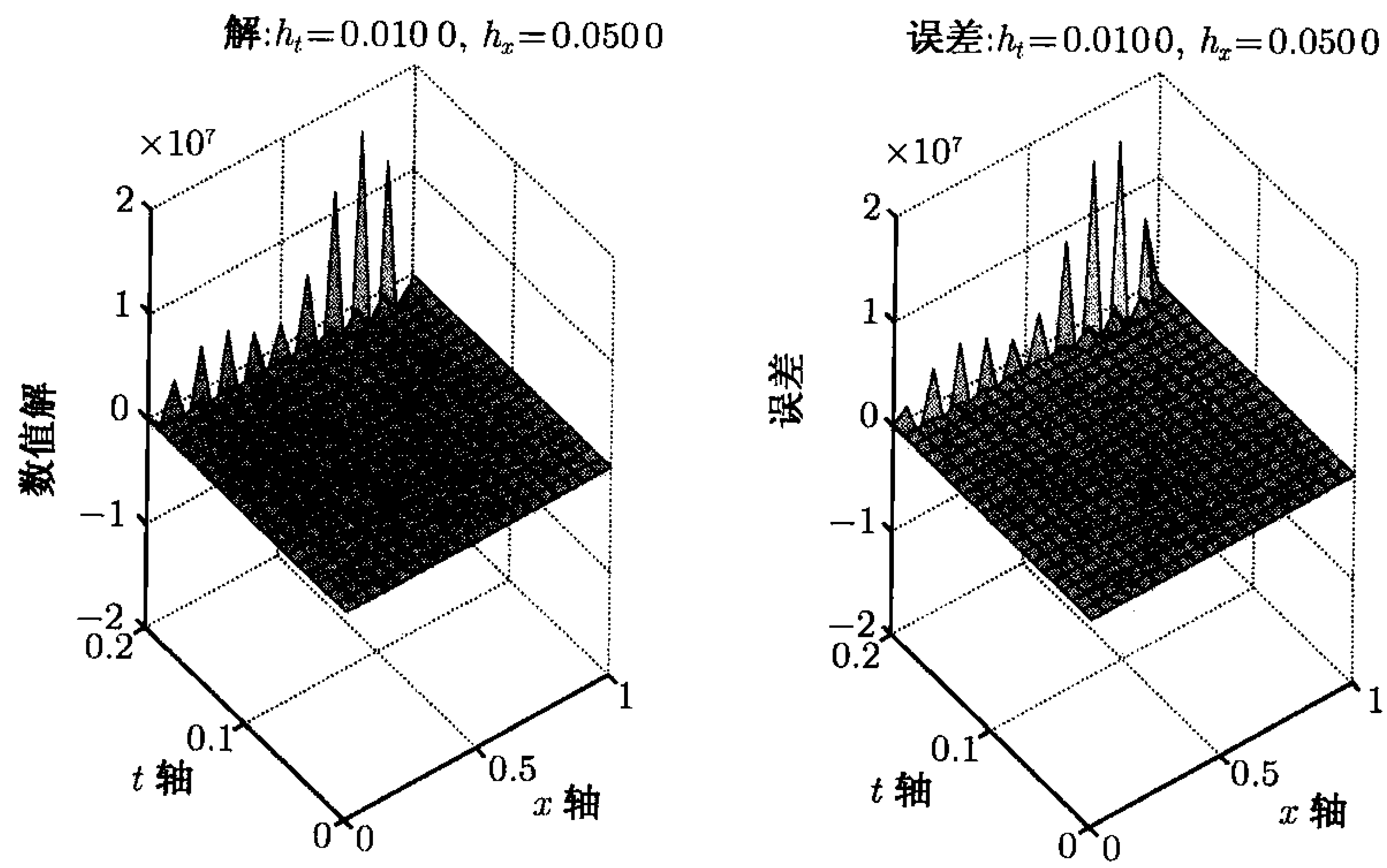


图 9-9 显式方法的数值结果: $n_x = 20$, $n_t = 20$

我们来探究是否存在始终保持稳定的数值方法的可能性. 用

$$\frac{\partial u}{\partial t}(x_i, t_k) \approx \frac{u_i^k - u_i^{k-1}}{h_t}$$

得到向后欧拉方法

$$\frac{u_i^k - u_i^{k-1}}{h_t} = a\frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{h_x^2} + f_i^k, \quad 2 \leqslant i \leqslant n_x$$

再次定义比值 $\gamma = ah_t/h_x^2$. 则数值格式为

$$u_i^1 = u_0(x_i) \quad 1 \leqslant i \leqslant n_x + 1 \tag{9.23}$$

并且对 $k \geqslant 2$,

$$\begin{cases} -\gamma u_{i-1}^k + (1+2\gamma)u_i^k - \gamma u_{i+1}^k = u_i^{k-1} + h_t f_i^k, & 2 \leqslant i \leqslant n_x \\ u_1^k = g_1(t_k), \quad u_{n_x+1}^k = g_2(t_k) \end{cases} \tag{9.24}$$

这里给定 $t=t_{k-1}$ 层上的近似解, 为求下一个时间层 $t=t_k$ 上的近似解必须解一个线性方程组. 这样的方法称为**隐式方法**.

利用比显式格式 (9.19~9.20) 的稳定性更细致的论证, 可以证明, 对任意的比值 γ, 隐式格式 (9.23~9.24) 能很好地控制时间向前推进时的误差传播. 我们称隐式格式是无条件稳定的.

类似于向前欧拉格式 (9.19~9.20), 当真解 u 充分光滑时可以证明下列向后欧拉格式 (9.23~9.24) 的误差界成立:

$$\max_{\substack{1\leqslant i\leqslant n_x+1\\1\leqslant k\leqslant n_t+1}}|u(x_i,t_k)-u_i^k|=O(h_t+h_x^2) \tag{9.25}$$

即格式具有 h_x 的 2 阶精度和 h_t 的 1 阶精度.

为了在每个时间层 $t=t_k(k\geqslant 2)$ 实施向后欧拉格式, 需要解线性方程组:

$$\boldsymbol{A}\boldsymbol{u}^k=\boldsymbol{F}^k$$

这里, 系数阵是

$$\boldsymbol{A}=\begin{bmatrix}1+2\gamma & -\gamma & & & \\ -\gamma & 1+2\gamma & -\gamma & & \\ & \ddots & \ddots & \ddots & \\ & & -\gamma & 1+2\gamma & -\gamma \\ & & & -\gamma & 1+2\gamma\end{bmatrix}_{(n_x-1)\times(n_x-1)}$$

未知向量 $\boldsymbol{u}^k=[u_2^k,\cdots,u_{n_x}^k]^{\mathrm{T}}$, 而右端向量

$$\boldsymbol{F}^k=[u_2^{k-1}+h_tf_2^k+\gamma g_1(t_k),\ u_3^{k-1}+h_tf_3^k,\cdots,u_{n_x-1}^{k-1}+h_tf_{n_x-1}^k,u_{n_x}^{k-1}+h_tf_{n_x}^k+\gamma g_2(t_k)]^{\mathrm{T}}$$

因为对任意的 k, 系数阵 $\boldsymbol{A}$ 是相同的, 所以只需计算它的 $\boldsymbol{LU}$ 分解一次, 然后就可以用 $\boldsymbol{LU}$ 分解求解所有的 k 阶线性方程组. 因为 $\boldsymbol{A}$ 是三对角阵, 所以可以使用 6.4.2 节三对角阵的 $\boldsymbol{LU}$ 分解的程序`tridiag`.

MATLAB 程序 下面是隐式方法的一个 MATLAB 程序.

```
function U = Heat2(a,f,u0,g1,g2,L,T,nx,nt)
%
% function U = Heat2(a,f,u0,g1,g2,L,T,nx,nt)
%
% The backward difference scheme for solving the initial boundary
% value problem of the heat equation u_t=a u_{xx}+f for x in
% [0,L], and t in [0,T].
```

```
% Input
%       a: the coefficient of the u_{xx} term
%       f=f(x,t): the right-hand-side function
%       u0=u0(x): the initial value function
%       g1=g1(t): the boundary value function at x=0
%       g2=g2(t): the boundary value function at x=L
%       L:  right end-point of the spatial interval
%       T:  right end-point of the time interval
%       nx: the number of subintervals of [0,L]
%       nt: the number of subintervals of [0,T]
% Output
%        U: the solution (u_{k,i}), k the time grid point index,
%           i the space grid point index
% compute some parameters
nx1 = nx+1;
hx = L/nx;
nt1 = nt+1;
ht = T/nt;
r = a*ht/(hx*hx)
r1 = 1+2*r;
%  generate the grid points for x and t variables
xvec = hx*(0:nx);
tvec = ht*(0:nt);
%  generate the entries of the coefficient matrix for the linear
%  tridiagonal systems to be solved at each time level
avec(2:nx-1) = -r;
bvec(1:nx-1) = r1;
cvec(1:nx-2) = -r;
%  define the dimensions of the unknown
U = zeros(nt1,nx1);
%  the initial value
U(1,:) = feval(u0,xvec(:)');
%  compute the solution at t=ht; use the program tridiag to
%  factor and store A=LU and solve the system at t=ht
U(2,1) = feval(g1,ht);
U(2,nx1) = feval(g2,ht);
fvec(1) = U(1,2)+ht*feval(f,xvec(2),tvec(2))+r*U(2,1);
fvec(2:nx-2) = U(1,3:nx-1)+ht*feval(f,xvec(3:nx-1),tvec(2));
fvec(nx-1) = U(1,nx)+ht*feval(f,xvec(nx),tvec(2))+r*U(2,nx1);
[U(2,2:nx),avec,bvec,ier] = tridiag(avec,bvec,cvec,fvec,nx-1,0);
%  compute the solution at the other time levels
for k = 2:nt1
  U(k,1) = feval(g1,(k-1)*ht);
  U(k,nx1) = feval(g2,(k-1)*ht);
  fvec(1) = U(k-1,2)+ht*feval(f,xvec(2),tvec(k))+r*U(k,1);
```

```
    fvec(2:nx-2) = U(k-1,3:nx-1)+ht*feval(f,xvec(3:nx-1),tvec(k));
    fvec(nx-1) = U(k-1,nx)+ht*feval(f,xvec(nx),tvec(k))+r*U(k,nx1);
    U(k,2:nx) = tridiag(avec,bvec,cvec,fvec,nx-1,1);
end
% plot the numerical solution
surf(xvec,tvec,U)
xlabel('x-axis')
ylabel('t-axis')
zlabel('the numerical solution')
s1 = sprintf('h_t=%6.4f  h_x=%6.4f',ht,hx)
title(s1)
```

例 9.2.3 利用向后欧拉方法 (9.23~9.24) 求解例 9.2.1 的问题. 图 9-10 中画出对应于 $(n_x, n_t) = (20, 20)$ 的数值结果. 我们看到不存在数值解的不稳定问题.

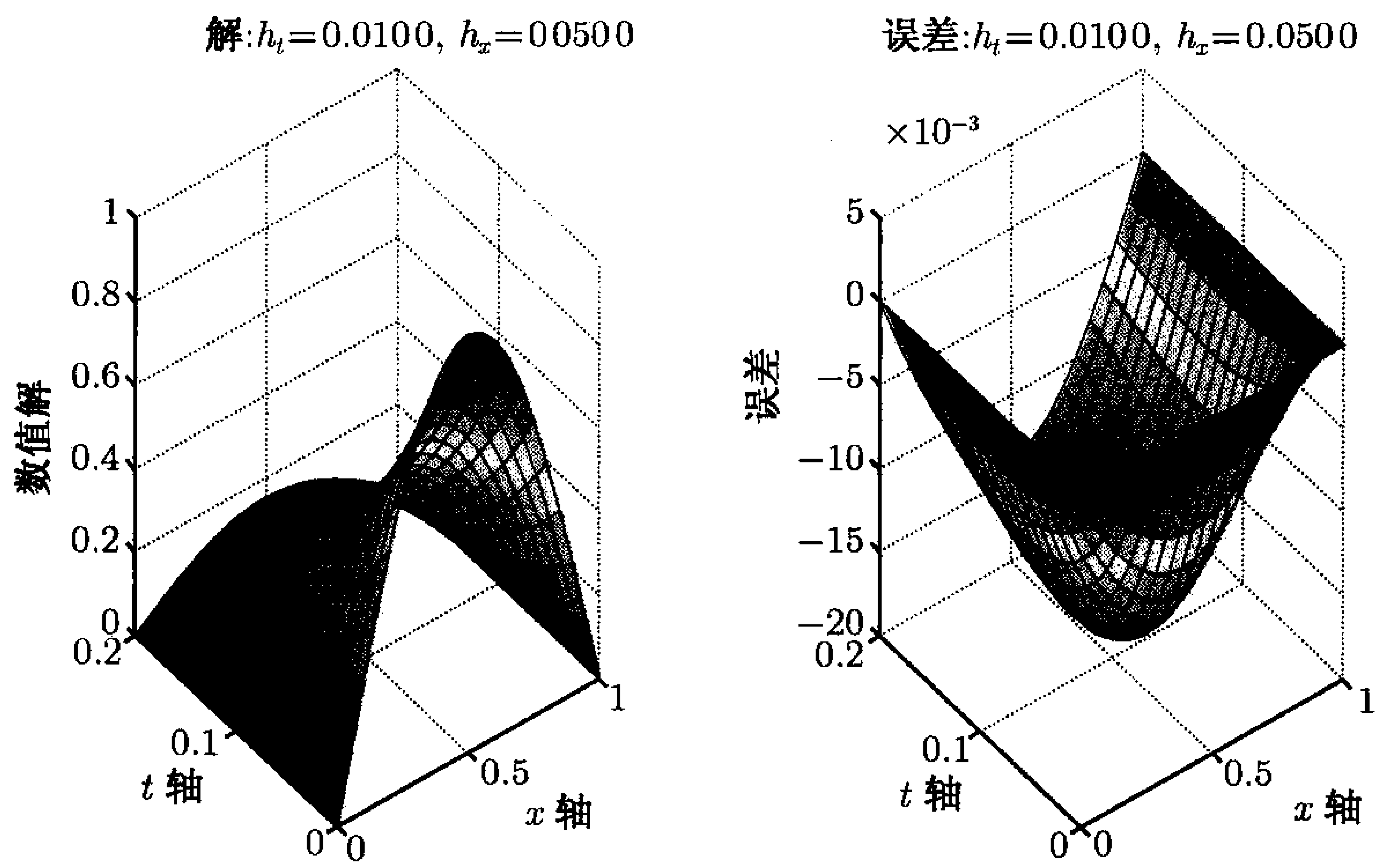


图 9-10 隐式方法的数值结果: $n_x = 20$, $n_t = 20$

为了更仔细地观察误差的性态, 我们对几对 n_x 和 n_t 计算数值解误差以及相关的比值 γ. 数值结果在表 9-5 中给出. 首先取 $n_t = n_x = 5$, 然后分别增加 n_t 和 n_x, 对 $n_t = n_x = 5$ 以及那些较大的 n_t 和 n_x 值计算数值解误差的比. 当 n_t 和 n_x 都加倍时, 误差近似地减少一半, 显示出线性收敛的性态. 当 n_x 加倍, 而 n_t 扩大 4 倍时, 数值解误差减少的因子接近 4(在表中是 3.82). 若让 n_t 和 n_x 同时扩大 4 倍, 则误差减少的因子还是接近于 4(在表中是 4.29). 当起始的 n_t 和 n_x 值较大时, 表 9-5 中的数值解误差的比会比 3.82 和 4.29 更靠近 4. 这个现象与表明方法有 1 阶 h_t 和 2 阶 h_x 的理论误差界 (9.25) 一致. 这样的方法, 当 n_t 加倍时, 应该让 n_x 扩大 4 倍才更为有效. ■

表 9-5 例 9.2.3 在 $t=2$ 处数值解误差

x	$n_t=5$ $n_x=5$	$n_t=10$ $n_x=10$	比率	$n_t=20$ $n_x=10$	比率	$n_t=20$ $n_x=20$	比率
0.2	3.50E−2	−1.67E−2	2.09	−9.15E−3	3.82	−8.15E−3	4.29
0.4	−5.66E−2	−2.70E−2	2.09	−1.48E−2	3.82	−1.32E−2	4.29
0.6	−5.66E−2	−2.70E−2	2.09	−1.48E−2	3.82	−1.32E−2	4.29
0.8	−3.50E−2	−1.67E−2	2.09	−9.15E−3	3.82	−8.15E−3	4.29

本节中对求解一维抛物型方程的初边值问题研究了两种差分方法. 这两种方法都有误差界 $O(h_t+h_x^2)$. 导出具有较高收敛阶的格式是可能的. 一个例子是具有误差界 $O(h_x^2+h_x^2)$ 的克兰克–尼克尔森(Crank-Nicholson) 格式. 关于这个格式和其他高阶差分格式, 请参阅 Morton and Mayers [25].

习题

1. 对任意常数 k, ω, c_1 和 c_2, 验证函数

$$u(x,t)=\mathrm{e}^{-k\omega^2t}(c_1\cos\omega x+c_2\sin\omega x)$$

满足微分方程

$$u_t=ku_{xx}$$

2. 证明: 用时间导数的向前欧拉差分离散 ODE 系统 (9.17) 可以导出显式全离散格式 (9.20).

3. 证明: 用时间导数的向后欧拉差分离散 ODE 系统 (9.17) 可以导出隐式全离散格式 (9.24).

4. 验证函数 $u(x,t)=\mathrm{e}^{-t}\sin x$ 是下列初边值问题的解

$$\begin{cases} u_t=u_{xx}, & 0<x<\pi,\quad t>0 \\ u(0,t)=u(\pi,t)=0, & t\geqslant 0 \\ u(x,0)=\sin x, & 0\leqslant x\leqslant\pi \end{cases}$$

选择几种 n_x 和 n_t, 利用程序`Heat1`和`Heat2`求解问题. 给出 $t=0.2$ 的最大数值解误差, 画出数值解及相应误差的图形.

5. 验证函数 $u(x,t)=\mathrm{e}^{-t}\cos x$ 是下列初边值问题的解

$$\begin{cases} u_t=u_{xx}, & 0<x<\pi,\quad t>0 \\ u_x(0,t)=u_x(\pi,t)=0, & t\geqslant 0 \\ u(x,0)=\cos x, & 0\leqslant x\leqslant\pi \end{cases}$$

注意, 这里是在空间导数值上指定边界条件. 这种边界条件称为诺伊曼 (Neumann) 边界条件.

6. 证明

$$u(x,t)=x^2-1+\mathrm{e}^{-t}\cos(\pi x/2)$$

是下列初边值问题的解

$$\begin{cases} u_t = \dfrac{4}{\pi^2}u_{xx} - \dfrac{8}{\pi^2}, & 0 < x < 1, \quad t > 0 \\ u_x(0,t) = 0, \quad u(1,t) = 0, & t \geqslant 0 \\ u(x,0) = x^2 - 1 + \cos(\pi x/2), & 0 \leqslant x \leqslant 1 \end{cases}$$

7. 用初值条件 $u(x,0)=0$ 和边界条件 $u(0,t)=t^2$, $u(1,t)=0$ 解方程 $u_t=u_{xx}$, $0<x<1$, $0<t\leqslant 1$. 对 $n_x=4,6,8$ 以及相应的 $n_t=16,36,64$, 利用显式格式 (9.19~9.20). 对 $x=1/4$, $1/2$, $3/4$ 和 $t=1$ 制作一张数值解表格. 画出数值解的图形.

8. 重复习题 7 但对 $n_x=n_t=4,6,8,16$ 利用隐式格式 (9.23~9.24).

9. 考虑均质的单位长杆的热传导问题. 假定杆是充分细的并且它的表面是绝热的. 则杆的温度 $u=u(x,t)$ 满足方程

$$u_t = au_{xx} + f(x,t)$$

其中 $a=k/(c\rho)$, c 是特定的热度, ρ 是密度, k 是热传导率系数, 而 $f(x,t)$ 是标度热源函数. 假如杆的两端温度固定为 0, 初始的杆温度为 0, $a=1$, 而 $f(x,t)=x(1-x)\sin\pi t$. 对几个 n_t 和 n_x 的值计算并画出在 $t=1$ 处的温度, 直到在 $x=1/2$, $t=1$ 处的数值解有三位稳定的数字.

10. 对习题 6 中求解的初边值问题给出一个差分格式. 类似于公式 (8.143) 进行边界条件 $u_x(0,t)=0$ 的离散化.

11. 推导一维抛物型方程的差分格式的思想可直接推广到高维抛物型方程. 例如, 在单位正方形 $(0,1)^2$ 上考虑二维抛物型方程

$$\frac{\partial u}{\partial t} = a\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

连同初始条件

$$u(x,y,0) = u_0(x,y)$$

和边界条件

$$\begin{aligned} u(0,y,t) = g_l(y,t), \quad u(1,y,t) = g_r(y,t), \quad 0 \leqslant y \leqslant 1, t \geqslant 0 \\ u(x,0,t) = g_b(x,t), \quad u(x,1,t) = g_t(x,t), \quad 0 \leqslant x \leqslant 1, t \geqslant 0 \end{aligned}$$

假如对 $t\leqslant T$ 求解问题. 把单位区间分割成 $n_x=n_y=n$ 等分, 区间 $[0,T]$ 分成 n_t 等分. 记 $h=1/n$, $h_t=T/n_t$, $x_i=(i-1)h$, $y_j=(j-1)h$, $t_k=(k-1)h_t$, $1\leqslant i,j\leqslant n+1$, $1\leqslant k\leqslant n_t+1$. 真解 $u(x_i,y_j,t_k)$ 的近似值用 u_{ij}^k 表示. 类似于显式方法 (9.19~9.20) 和隐式方法 (9.23~9.24), 对 $\{u_{ij}^k\}$ 建立有限差分方程组.

9.3 一维波动方程

最简单的二阶双曲型方程是

$$\frac{\partial^2 u}{\partial t^2} = a\frac{\partial^2 u}{\partial x^2} + f \tag{9.26}$$

这个方程可描述一维波的传播以及许多其他的物理过程. 设方程的空间变量范围为 $x \in (0, L)$. 为了得到唯一解, 需要在 $x = 0$ 和 $x = L$ 处指定边界条件并在 $t = 0$ 处指定初始条件. 这里, 考虑如下的初边值条件:

$$
\begin{aligned}
u(0,t) = & g_1(t), \quad u(L,t) = g_2(t), \quad t \geqslant 0 & (9.27)\\
u(x,0) = & u_0(x), \quad u_t(x,0) = v_0(x), \quad 0 \leqslant x \leqslant L & (9.28)
\end{aligned}
$$

因为方程关于时间是 2 阶的, 所以需要两个初始条件. 微分方程 (9.26)、边界条件 (9.27) 和初始条件 (9.28) 合在一起构成一个初边值问题. 边界条件和初始条件之间的相容性要求

$$u_0(0) = g_1(0), \quad u_0(L) = g_2(0)$$

我们对 $0 \leqslant t \leqslant T$ 来求解初边值问题. 类似于上一节求解抛物型方程问题, 这里我们也可以考虑两种数值方法. 第一种方法引入半离散化把问题转换成一个 ODE 方程组然后用 ODE 解算器求解. 第二种方法同时离散空间变量和时间变量得到一个全离散格式. 下面考虑初边值问题的一个常见的全离散格式.

把 $[0, T]$ 分割成 n_t 等分, 时间步长为 $h_t = T/n_t$. 把 $[0, L]$ 分割成 n_x 等分, 空间步长为 $h_x = L/n_x$. 记 $x_i = (i-1)h_x, 1 \leqslant i \leqslant n_x + 1, t_k = (k-1)h_t, 1 \leqslant k \leqslant n_t + 1$. 用 u_i^k 表示解 $u(x_i, t_k)$ 的近似值. 用 2 阶中心差分逼近 $u_{tt}(x_i, t_k)$ 和 $u_{xx}(x_i, t_k)$, 则在内部网格点 (x_i, t_k) 处的显式中心差分方程是

$$\frac{u_i^{k+1} - 2u_i^k + u_i^{k-1}}{h_t^2} = a\frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{h_x^2} + f_i^k \tag{9.29}$$

格式的截断误差定义为差分方程 (9.29) 的左端减去右端, 其中数值解值用对应网格点上的真解代替. 因为

$$\frac{u(x_i, t_{k+1}) - 2u(x_i, t_k) + u(x_i, t_{k-1})}{h_t^2}$$

是 $u_{tt}(x_i, t_k)$ 的 $O(h_t^2)$ 逼近, 而

$$\frac{u(x_{i+1}, t_k) - 2u(x_i, t_k) + u(x_{i-1}, t_k)}{h_x^2}$$

是 $u_{xx}(x_i, t_k)$ 的 $O(h_x^2)$ 逼近, 所以格式的截断误差是 $O(h_x^2 + h_t^2)$.

作为一个完整的格式, 内部网格点处的方程 (9.29) 还得补充数值边值条件和初值条件. 边界条件来自 (9.27):

$$u_1^k = g_1(t_k), \quad u_{n_x+1}^k = g_2(t_k), \quad 1 \leqslant k \leqslant n_t + 1 \tag{9.30}$$

关于初始条件, 利用

$$u_i^1 = u_0(x_i), \quad 1 \leqslant i \leqslant n_x + 1 \tag{9.31}$$

$$\frac{u_i^2 - u_i^0}{2h_t} = v_0(x_i), \quad 1 \leqslant i \leqslant n_x + 1 \tag{9.32}$$

这里新的变量 u_i^0, $1 \leqslant i \leqslant n_x + 1$, 是真解 u 被适当地扩充到负的 t 时 $u(x_i, -h_t)$ 的近似值; 扩充的解对负的 t 满足同样的方程 (9.26). 引入新的变量后, 可以使用差商

$$\frac{u(x_i, h_t) - u(x_i, -h_t)}{2h_t}$$

作为导数 $u_t(x_i, 0)$ 的 $O(h_t^2)$ 逼近. 逼近初始速度条件 $u_t(x, 0) = v_0(x)$ 的一个简单方法是

$$\frac{u_i^2 - u_i^1}{h_t} = v_0(x_i), \quad 1 \leqslant i \leqslant n_x + 1$$

然而, 这个逼近导致较低收敛阶的数值方法, 这里不使用它.

由于引入了新的变量 $u_i^0 (1 \leqslant i \leqslant n_x + 1)$, 我们需要在 $(x_i, 0)(2 \leqslant i \leqslant n_x)$ 处的差分方程. 规定差分方程 (9.29) 对 $k = 1$(即 $t = 0$) 也成立. 记 $\gamma = a h_t^2 / h_x^2$. 则从 (9.29) 和 (9.32) 分别得到下列方程:

$$u_i^2 = \gamma u_{i-1}^1 + 2(1 - \gamma) u_i^1 + \gamma u_{i+1}^1 - u_i^0 + h_i^2 f_i^1$$

$$u_i^2 = u_i^0 + 2h_t v_0(x_i)$$

两个方程相加可消去 u_i^0,

$$2u_i^2 = \gamma u_{i-1}^1 + 2(1 - \gamma) u_i^1 + \gamma u_{i+1}^1 + 2h_t v_0(x_i) + h_t^2 f_i^1$$

这样,

$$\begin{cases} u_1^2 = g_1(h_t), \quad u_{n_x+1}^2 = g_2(h_t) \\ u_i^2 = \dfrac{\gamma}{2} u_{i-1}^1 + (1 - \gamma) u_i^1 + \dfrac{\gamma}{2} u_{i+1}^1 + h_t v_0(x_i) + \dfrac{h_t^2}{2} f_i^1, \\ \hfill 2 \leqslant i \leqslant n_x \end{cases} \tag{9.33}$$

总结如下: 首先, 利用 (9.31) 计算 $\{u_i^1\}_{i=1}^{n_x+1}$, 并利用 (9.33) 计算 $\{u_i^2\}_{i=1}^{n_x+1}$, 然后, 对 $k = 2, \cdots, n$ 计算

$$\begin{cases} u_1^{k+1} = g_1(kh_t), \quad u_{n_x+1}^{k+1} = g_2(kh_t) \\ u_i^{k+1} = \gamma u_{i-1}^k + 2(1 - \gamma) u_i^k + \gamma u_{i+1}^k - u_i^{k-1} + h_t^2 f_i^k, \\ \hfill 2 \leqslant i \leqslant n_x \end{cases} \tag{9.34}$$

可以证明稳定性条件是 $\gamma \leqslant 1$, 即

$$\sqrt{a}h_t \leqslant h_x$$

这个条件与用显式格式解抛物型方程时的稳定性条件 (9.21) 相比, 限制比较小.

当真解 u 充分光滑时, 误差界的理论结果是

$$\max_{\substack{1\leqslant i\leqslant n_x+1\\1\leqslant k\leqslant n_t+1}} |u(x_i,t_k)-u_i^k| = O(h_t^2+h_x^2) \tag{9.35}$$

即当边界条件的逼近至少有相同阶的精度时, 该方法的阶与截断误差的阶相同. 故格式关于 x 步长和 t 步长都是 2 阶的.

MATLAB 程序 实施上面格式的程序如下:

```
function U = Wave(a,f,u0,v0,g1,g2,L,T,nx,nt)
%
% function U=Wave(a,f,u0,v0,g1,g2,L,T,nx,nt)
%
% The centered difference scheme for solving the initial boundary
% value problem of the wave equation u_{tt}=a u_{xx}+f for
% x in [0,L], and t in [0,T].
% Input
%      a: the coefficient of the u_{xx} term
%      f=f(x,t): the right-hand-side function
%      u0=u0(x): the initial value function
%      v0=v0(x): the initial derivative value function
%      g1=g1(t): the boundary value function at x=0
%      g2=g2(t): the boundary value function at x=L
%      L:  right end-point of the spatial interval
%      T:  right end-point of the time interval
%      nx: the number of subintervals of [0,L]
%      nt: the number of subintervals of [0,T]
% Output
%       U: the solution (u_{k,i}), k the time grid point index,
%          i the space grid point index
% compute some parameters
nx1 = nx+1;
hx = L/nx;
nt1 = nt+1;
ht = T/nt;
ht2 = ht*ht;
r = a*ht*ht/(hx*hx)
r1 = 2*(1-r);
if r > 1
  disp('The ratio is too big; the scheme is unstable!')
end
```

```
% define the grid points
xvec = hx*(0:nx);
tvec = ht*(0:nt);
% compute the solution at t=0 and ht using the initial values
U = zeros(nt1,nx1);
U(1,:) = feval(u0,xvec);
U(2,1) = feval(g1,ht);
U(2,2:nx) = 0.5*(r*(U(1,1:nx-1)+U(1,3:nx+1))+r1*U(1,2:nx) ...
           +ht2*feval(f,xvec(2:nx),tvec(1))) ...
           +ht*feval(v0,xvec(2:nx));
U(2,nx1) = feval(g2,ht);
% compute the solution at the other times
for k = 2:nt
  U(k+1,1) = feval(g1,k*ht);
  U(k+1,nx1) = feval(g2,k*ht);
  U(k+1,2:nx) = r*(U(k,1:nx-1)+U(k,3:nx+1))+r1*U(k,2:nx) ...
              -U(k-1,2:nx)+ht2*feval(f,xvec(2:nx),tvec(k));
end
% plot the numerical solution
surf(xvec,tvec,U)
xlabel('x-axis')
ylabel('t-axis')
zlabel('the numerical solution')
s1=sprintf('h_t=%6.4f  h_x=%6.4f',ht,hx)
title(s1)
```

例 9.3.1　求解下列初边值问题:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial^2 u}{\partial x^2} + 2\mathrm{e}^{-t}\sin x, & x \in (0,\pi), t \in (0,1) \\ u(x,0) = \sin x, \quad u_t(x,0) = -\sin x, & x \in [0,\pi] \\ u(0,t) = u(\pi,t) = 0, & t \in [0,1] \end{cases}$$

真解是 $u(x,t) = \mathrm{e}^{-t}\sin x$, 如图 9-11 所示.

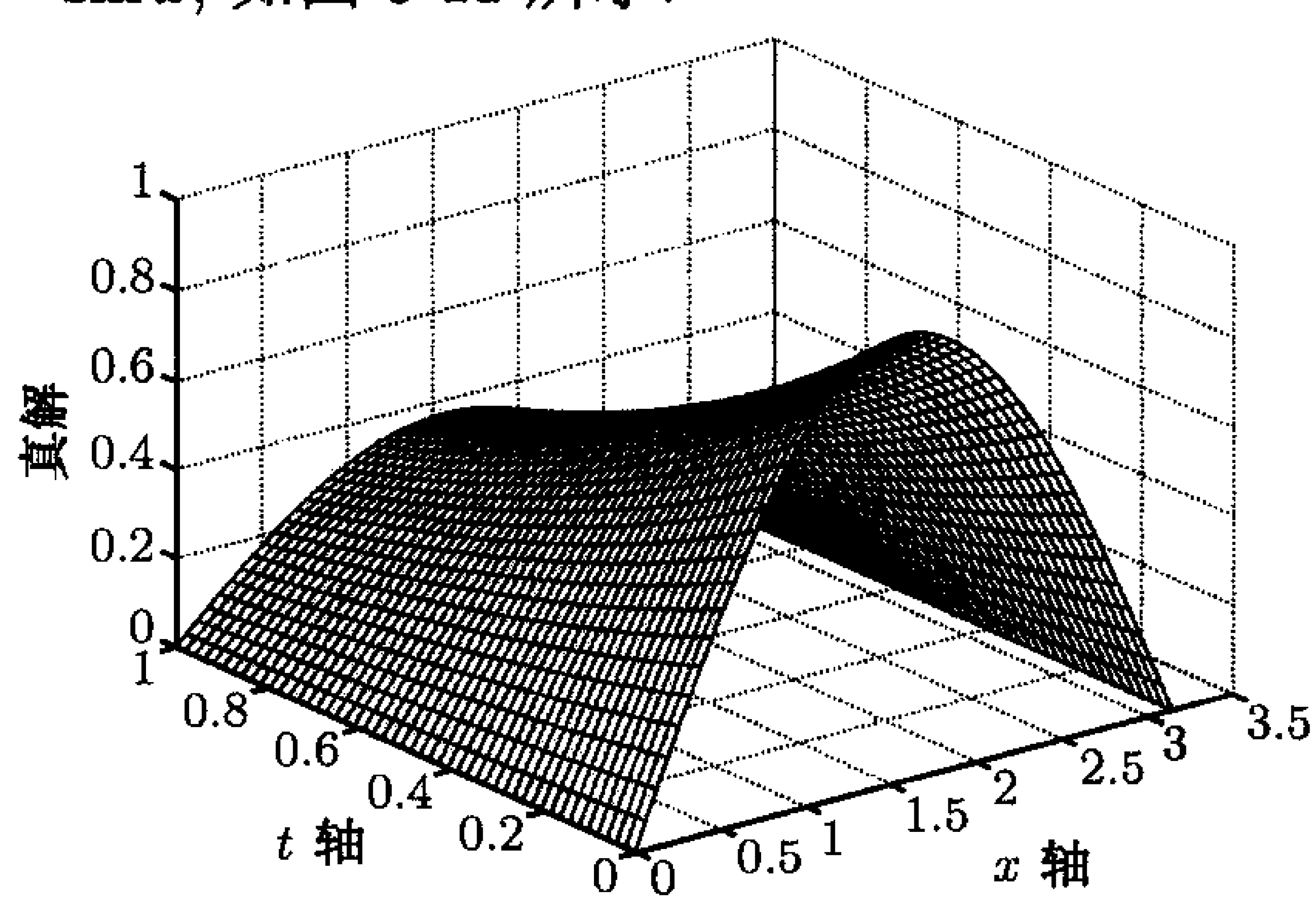


图 9-11　例 9.3.1 的真解

对 $(n_x, n_t) = (5,5), (10,10), (20,20)$ 求解此问题. 因为本例中真解是已知的, 所以可以计算数值解的误差. 对应于 $(n_x, n_t) = (20,20)$ 的数值结果如图 9-12 所示.

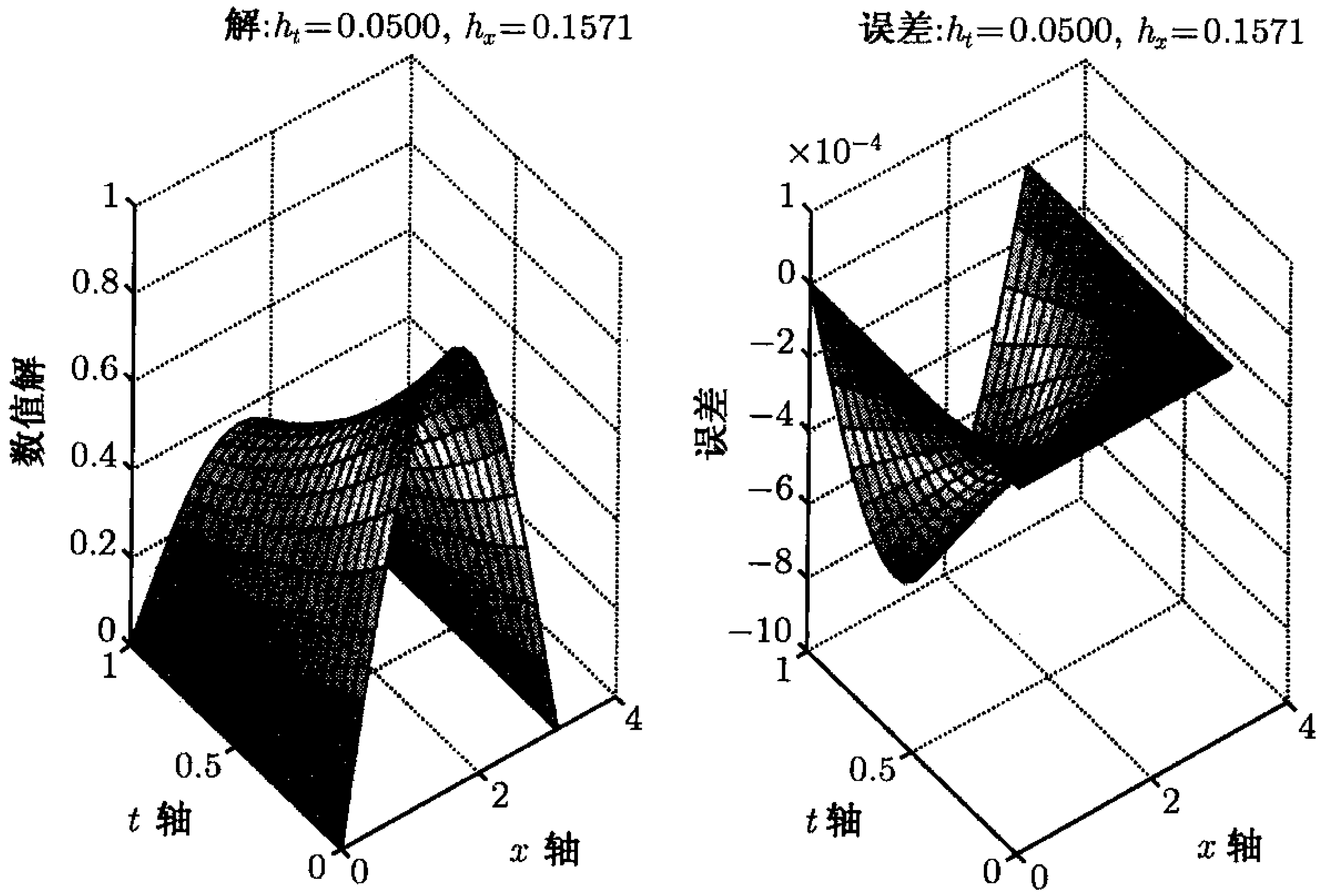


图 9-12 $n_x = 20$ 和 $n_t = 20$ 时的数值结果

为更清楚地观察误差性态, 表 9-6 给出 $t_k = 0.2, 0.4, 0.6, 0.8, 1$ 时的最大的数值解误差 $\max_{1\leqslant i\leqslant n_x+1}\left|u(x_i,t_k)-u_i^k\right|$. 这些比值都接近于 4, 这表明方法的收敛阶为 2.

表 9-6 例 9.3.1 的最大数值解误差

t	$n=5$	$n=10$	比率	$n=20$	比率
0.2	1.824 7E−3	4.717 5E−4	3.87	1.174 4E−4	4.02
0.4	4.486 1E−3	1.166 0E−3	3.85	2.906 6E−4	4.01
0.6	7.716 8E−3	2.011 6E−3	3.84	5.018 2E−4	4.01
0.8	1.125 7E−2	2.939 2E−3	3.83	7.335 4E−4	4.01
1.0	1.486 0E−3	3.882 9E−3	3.83	9.692 5E−4	4.01

表 9-7 给出 $t=1$ 和 $x=\pi/5$, $2\pi/5$, $3\pi/5$ 和 $4\pi/5$ 处的数值解误差以及相应的比率. 这些比值都接近于 4, 也与方法为 2 阶的理论结果一致. ■

表 9-7 例 9.3.1 在 t=1 处的数值解误差

x	$n=5$	$n=10$	比率	$n=20$	比率
$\pi/5$	−9.18E−3	−2.28E−3	4.02	−5.70E−4	4.01
$2\pi/5$	−1.49E−2	−3.69E−3	4.02	−9.22E−4	4.01
$3\pi/5$	−1.49E−2	−3.69E−3	4.02	−9.22E−4	4.01
$4\pi/5$	−9.18E−3	−2.28E−3	4.02	−5.07E−4	4.01

习题

1. 设 F 和 G 是二次可微函数. 证明函数

$$u(x,t)=F(x+\sqrt{a}\ t)+G(x-\sqrt{a}\ t)$$

满足方程

$$u_{tt}=au_{xx}$$

2. 考虑实直线 $x\in\mathbb{R}$ 上的齐次波动方程

$$u_{tt}=au_{xx}$$

因为空间域在两个方向无界, 所以只需指定初始条件

$$u(x,0)=u_0(x),\quad u_t(x,0)=v_0(x),\quad x\in\mathbb{R}$$

验证解的公式

$$\begin{aligned}u(x,t)&=\frac{1}{2}\left[u_0(x+\sqrt{a}\ t)+u_0(x-\sqrt{a}\ t)\right]+\frac{1}{2\sqrt{a}}\int_{x-\sqrt{a}\ t}^{x+\sqrt{a}\ t}v_0(s)\mathrm{d}s\\&\equiv u_1(x,t)+u_2(x,t)\end{aligned}$$

此式称为达朗贝尔 (D'Alembert) 公式. 解的 $u_1(x,t)$ 部分代表对初速度为零时的初始位移的传播, 而 $u_2(x,t)$ 部分描述由具有零初始位移的初始速度所引起的振动. 量 $\sqrt{a}$ 可解释为波传播速率.

3. 考虑两个 1 阶方程的方程组

$$\begin{aligned}u_t(x,t)&=v_x(x,t)+f_1(x,t)\\v_t(x,t)&=u_x(x,t)+f_2(x,t)\end{aligned}$$

其中 $f_1(x,t)$ 和 $f_2(x,t)$ 是可微函数, 证明 u 和 v 各满足一个形如 (9.26) 的波动方程.

4. 弦的振动运动用下列波动方程描述

$$u_{tt}=\frac{1}{2}u_{xx},\quad 0<x<1, t>0$$

假如固定弦的两端, 弦的初始状态是静止的, 初始位置是

$$u(x,0)=\mathrm{e}^x\sin(\pi x),\quad 0<x<1$$

对 $t=1$ 和 $n_x=n_t=5,10,20$ 计算并画出位移 $u(x,t)$ 的有限差分近似值.

5. 当初始位置条件用条件

$$u(x,0)=\frac{1}{4}\min(x,1-x)=\begin{cases}\dfrac{x}{4}, & 0\leqslant x\leqslant\dfrac{1}{2}\\[2mm]\dfrac{1-x}{4}, & \dfrac{1}{2}<x\leqslant 1\end{cases}$$

替代时重复习题 4.

6. 对下列数据的改变重复习题 4: 初始位移和初始速度都假定为零, 弦的左端固定, 而在右端, $u(1,t) = 0.1\sin(\pi t)$.

7. 考虑平面上长度为 L 的弹性弦小横向振动问题, 在 x 轴上它的两端点 $x = 0$ 和 $x = L$ 是固定的. 设弦受单位质量密度 $f(x,t)$ 的垂直力的作用. 假定位移向量垂直于 x 轴且位移很小. 则关于垂直位移函数 $u(x,t)$ 的方程是

$$u_{tt} = au_{xx} + f(x,t)$$

其中 $a = T_0/\rho$, T_0 是弦的张力, 而 ρ 是弦的质量密度, 两者都假定为常量. 取 $L = 1$, $a = 1$. 对下列情况计算垂直位移:

(a) $f(x,t) = \sin\pi x, u(x,0) = 0$ 和 $u_t(x,0) = 0$

(b) $f(x,t) = 0, u(x,0) = 0$ 和 $u_t(x,0) = 1$

8. 仿照逼近 $u_t(x_i,0)$ 的差分公式 (9.32) 的做法对 $u_x(0,t_k)$ 和 $u_x(L,t_k)$ 的逼近导出类似的差分公式. 考虑初边值问题 (9.26) 和 (9.28), (9.27) 用下式代替

$$u_x(0,t) = g_1(t), \quad u(L,t) = g_2(t)$$

导出相应的有限差分方程组.

9. 修正求解初边值问题

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial^2 u}{\partial x^2} + \left(1 + \dfrac{\pi^2}{4}\right)\mathrm{e}^{-t}\cos\dfrac{\pi x}{2}, & x \in (0,1), t \in (0,1) \\ u(x,0) = \cos\dfrac{\pi x}{2}, \quad u_t(x,0) = -\cos\dfrac{\pi x}{2}, & x \in [0,1] \\ u_x(0,t) = u(1,t) = 0, & t \in [0,1] \end{cases}$$

的 MATLAB 程序`Wave`. 问题的真解是 $u(x,t) = \mathrm{e}^{-t}\cos(\pi x/2)$. 对 $n_x = n_t = 5, 10, 20$ 计算数值解并画出误差函数. 给出 $t = 1$ 和 $x = i/5(0 \leqslant i \leqslant 4)$ 处的解的误差, 并讨论所得到的数值结果.

附录A　中 值 定 理

本附录讨论中值定理. 只对应用感兴趣的人可能认为理解这些定理是不必要的; 但是, 事实上, 它们在推导数值方法、得到误差估计以及简化复杂的表达式方面非常有用.

中值定理讨论区间 $[a,b]$ 上的函数 $f(x)$. 这些定理断言, 在 $[a,b]$ 内存在特殊点 $x=c$ 使 $f(x)$ 的某个性质成立. 由于大多数微积分课本中对这些定理作了完整的讨论, 因此我们仅叙述这些定理并说明它们的用途.

定理 A.1(介值定理)　设 $f(x)$ 是区间 $a \leqslant x \leqslant b$ 上的连续函数. 记

$$M = \max_{a \leqslant x \leqslant b} f(x), \quad m = \min_{a \leqslant x \leqslant b} f(x)$$

则对每个满足 $m \leqslant v \leqslant M$ 的值 v, 在 $[a,b]$ 内至少存在一个点 c 使 $f(c)=v$.

这是一个直观的结论. 例如, 考虑整个白天温度 $T(t)$ 的变化, 其中 $T(t)$ 表示在时间 t 处的温度. 一般地可以认为 $T(t)$ 是 t 的连续函数. 因此, 如果 T 在早上 7:30 是 45°F, 而在下午 3:00 是 83°F, 那么在早上 7:30 与下午 3:00 之间, 45°F 与 83°F 之间的每一个温度至少出现一次. $T(t)$ 的一个可能的图形在图 A-1 中给出.

例 A.2　求 $0 \leqslant x \leqslant \pi$ 中 c 的值使得

$$\sin(c) = \frac{1}{2}$$

$\sin(x)$ 的图形在图 A-2 中给出. 由于在 $[0,\pi]$ 上, $\sin(x)$ 的最小值是 $m=0$ 而最大值是 $M=1$, 因此介值定理保证该方程有一个解 c. 根据函数 $\sin(x)$ 的知识, 容易证明 c 是 $\pi/6$ 和 $5\pi/6$. ■

介值定理的一个重要的作用是简化复杂的表示式. 作为较一般表示式的特例, 考虑

$$S = \frac{1}{3}[f(x_1)+f(x_2)+f(x_3)]$$

其中 $f(x)$ 在区间 $[a,b]$ 上连续并且 x_1, x_2, x_3 是区间 $[a,b]$ 内的点. 使用定理 A.1 的记号, 有

$$S \leqslant \frac{1}{3}[M+M+M] = M$$

并且

$$S \geqslant \frac{1}{3}[m+m+m] = m$$

因此, S 为区间 $[m,M]$ 内的一个值; 根据定理 A.1, 在区间内存在一个点 c 使得 $S=f(c)$, 即对于 $[a,b]$ 内的某个 c,

$$\frac{1}{3}[f(x_1)+f(x_2)+f(x_3)]=f(c) \tag{A.1}$$

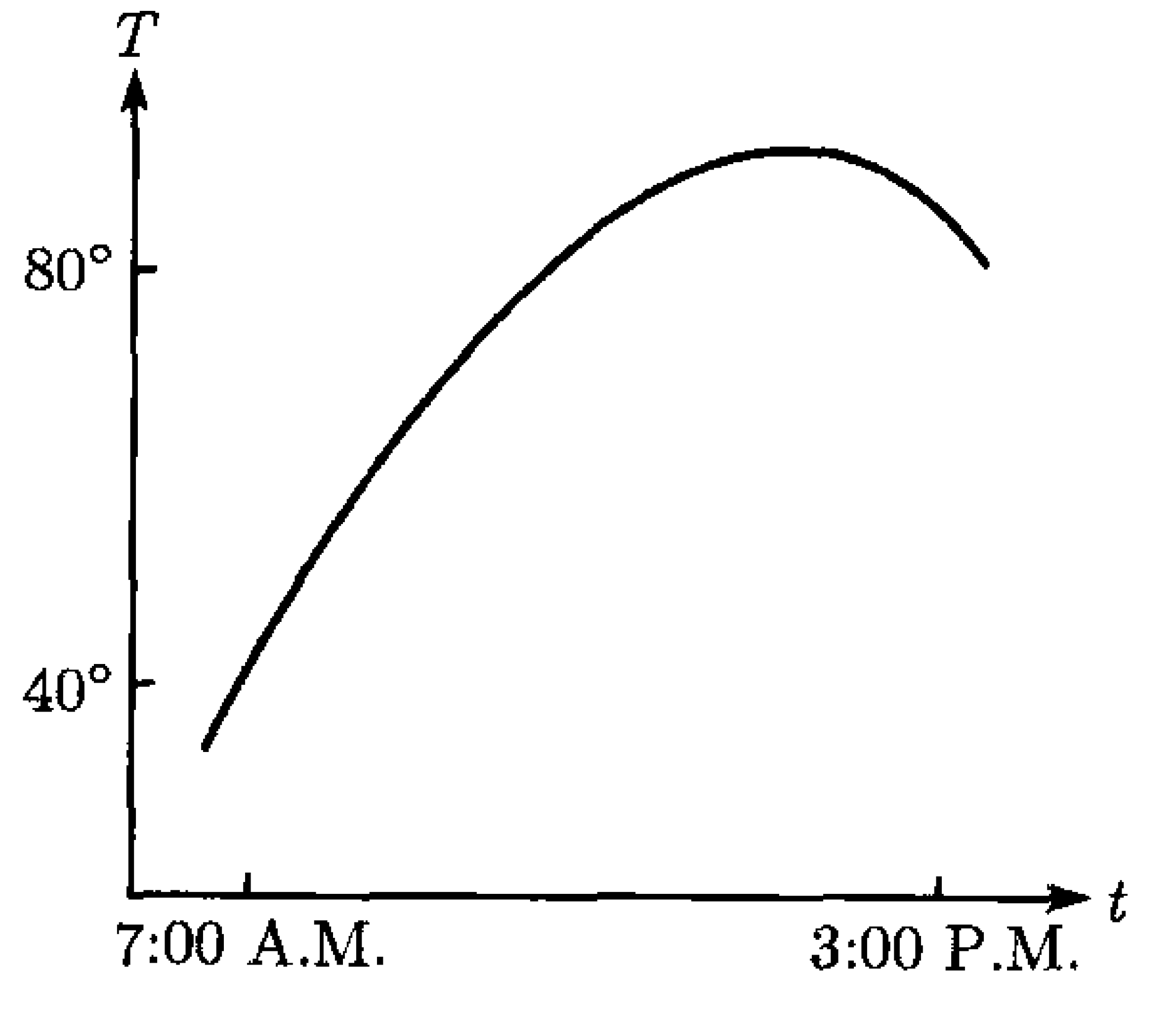


图 A-1 温度与时间关系

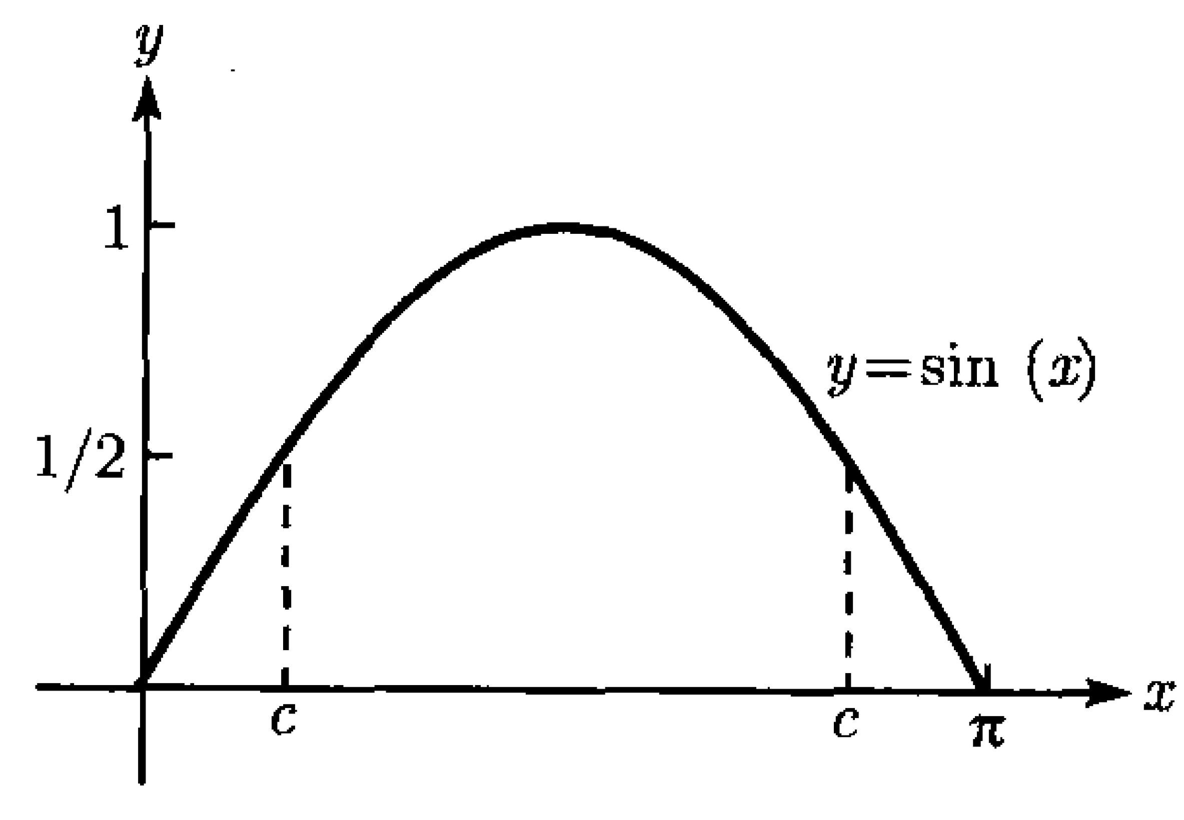


图 A-2 介值定理的说明

设 $f(x)$ 在 $[a,b]$ 上连续并且设 $x_1,x_2,\cdots,x_n$ 是 $[a,b]$ 内的点. 设 $w_1,w_2,\cdots,w_n$ 是满足

$$\sum_{j=1}^{n} w_j = 1$$

的正数. 定义

$$S=\sum_{j=1}^{n} w_j f(x_j)$$

则利用类似的论证可知, 在 $[a,b]$ 内存在一个点 c 使得 $S=f(c)$, 即对于 $[a,b]$ 内的某个 c,

$$\sum_{j=1}^{n} w_j f(x_j)=f(c) \tag{A.2}$$

例 A.3 设 $f(x)=\sqrt{x}$, 考虑

$$S=\frac{1}{5}[f(0)+f(0.25)+f(0.5)+f(0.75)+f(1)]$$

这里有 $n=5$ 个点, 并且所有 $w_j=1/5$. 我们来求满足 (A.2) 的值 c. 首先, $S\doteq 0.614\ 63$. 因此, 我们要求

$$\sqrt{c}\doteq 0.614\ 63$$

$$c\doteq (0.614\ 63)^2\doteq 0.377\ 8$$

称 S 为函数 $\sqrt{x}$ 在 $[0,1]$ 上的 "平均值" 或 "中值", 这个平均值是函数在 c 处的值. ■

定理 A.4(中值定理) 设 $f(x)$ 在区间 $a \leqslant x \leqslant b$ 上连续并且在 $a < x < b$ 上可微. 则在 (a, b) 内至少存在一个点 c, 使得

$$f(b) - f(a) = f'(c)(b - a) \tag{A.3}$$

例 A.5 对 [0, 2] 上的函数 $f(x) = \sqrt{x}$, 求满足 (A.3) 的 c 值. 图 A-3 是这种情况下定理的一个说明. 虚线连接点 $(0, 0)$ 和 $(2, \sqrt{2})$. 由 (A.3),

$$\sqrt{2} - \sqrt{0} = \frac{1}{2\sqrt{c}}(2 - 0)$$

这个方程有解 $c = 1/2$. 图形给出了经过点 $(\frac{1}{2}, f(\frac{1}{2}))$ 处的切线. ■

如果将 (A.3) 改写为

$$\frac{f(b) - f(a)}{b - a} = f'(c)$$

我们就可以给出这个定理的一个简单的几何解释. 连接图形端点的直线与曲线在某个点 $(c, f(c))$ 处的切线有同样的斜率. 用图 A-3 验证这个结论.

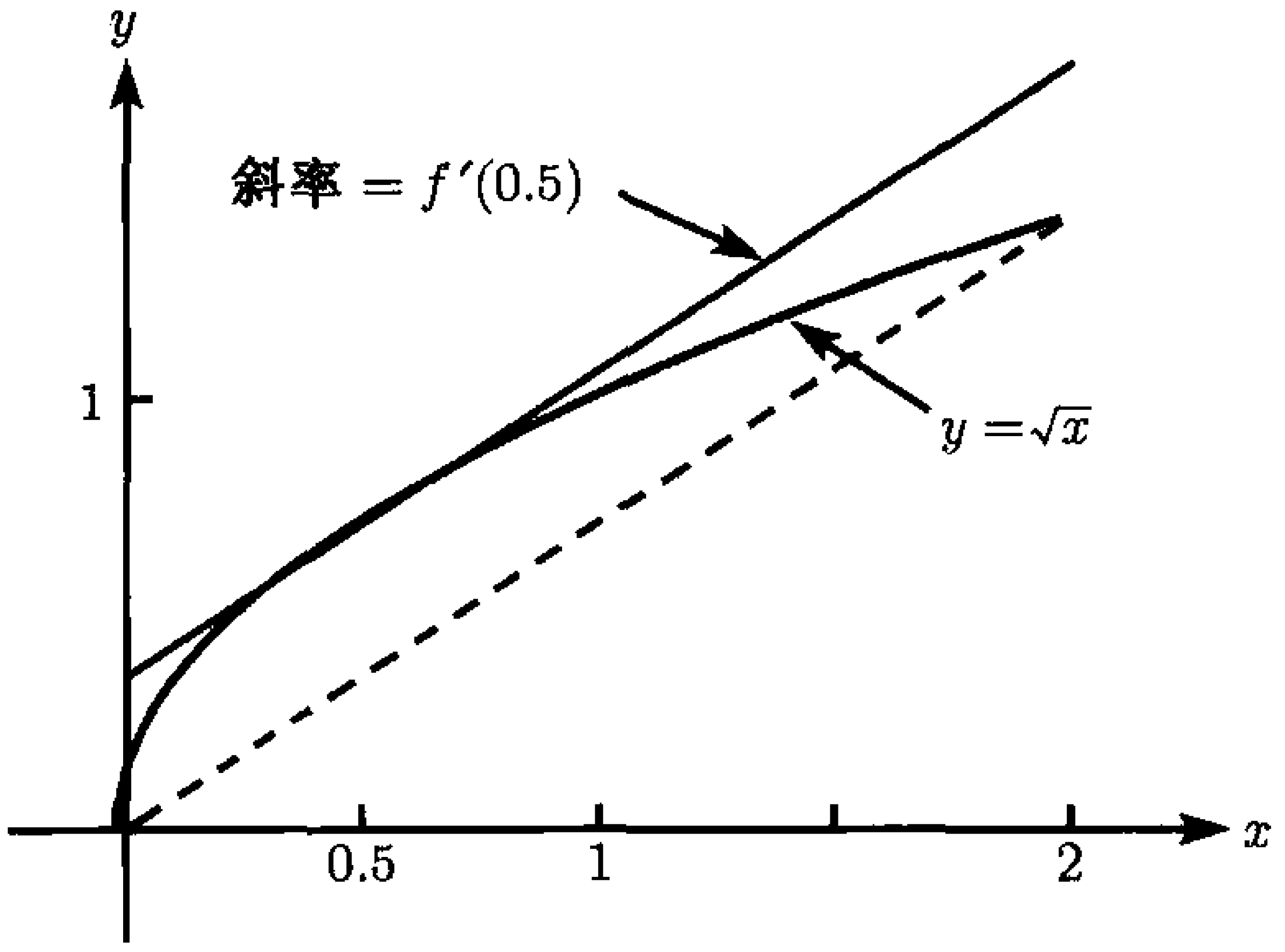


图 A-3 中值定理的例

(A.3) 的一个作用是估计或界定自变量的误差对函数误差的影响. 设 x_T 表示自变量的真值, 通常未知, 并设 x_A 表示一个已知的近似值. 则对于 x_T 和 x_A 之间的某个数 c,

$$f(x_T) - f(x_A) = f'(c)(x_T - x_A)$$

由于 x_T 和 x_A 一般非常接近, 对 x_T 和 x_A 之间的 x, $f'(x)$ 的值通常变化不大. 因此,

$$f(x_T) - f(x_A) \approx \begin{cases} f'(x_T)(x_T - x_A) \\ \quad\text{或} \\ f'(x_A)(x_T - x_A) \end{cases} \tag{A.4}$$

对具体的问题可以选择使用起来较方便的公式.

例 A.6 考虑使用 $\pi/6 \doteq 0.524$ 来计算 $\cos(\pi/6)$ 的值. 为了应用 (A.4), 取 $x_T = \pi/6$, $x_A = 0.524$, $f(x) = \cos(x)$. 一般来说, $x_T - x_A$ 未知而需要使用它的一个界. 但这儿为了说明方便, 我们用 $x_T - x_A \doteq -0.000\,4$. 利用 (A.4) 得到

$$\cos\left(\frac{\pi}{6}\right) - \cos(0.524) \doteq -\sin\left(\frac{\pi}{6}\right)\left(\frac{\pi}{6} - 0.524\right)$$
$$\doteq -\frac{1}{2}(-0.000\,4) = 0.000\,2$$

而, 实际误差是 0.000 200 7. ■

中值定理在得到函数因自变量变化而变化的界限方面也非常有用. 如果 $|f'(x)|$ 在区间 $a \leqslant x \leqslant b$ 上处处以常数 M 为上界, 则对 $[a, b]$ 内所有的 x, z, 有

$$|f(x) - f(z)| \leqslant M|x - z| \tag{A.5}$$

例 A.7 考虑 $f(x) = \mathrm{e}^x$, $0 \leqslant x \leqslant 1$. 则 $f'(x)$ 的上界为 e^1, 因此

$$|\mathrm{e}^x - \mathrm{e}^z| \leqslant e\,|x - z|, \quad 0 \leqslant x, z \leqslant 1$$ ■

还可以一些不同于这里描述的方式使用中值定理, 这些方式出现在本书中好几个地方. 中值定理是 1.1 节和 1.2 节描述的泰勒定理的特殊情况.

定理 A.8(积分中值定理) 设 $w(x)$ 是 (a, b) 上的非负可积函数, 并设 $f(x)$ 在 $[a, b]$ 上连续. 则在 $[a, b]$ 内至少存在一个点 c 使得

$$\int_a^b f(x)w(x)\mathrm{d}x = f(c)\int_a^b w(x)\mathrm{d}x \tag{A.6}$$

特别地, 如果取 $w(x) \equiv 1$, 则对 $[a, b]$ 中某个 c

$$\int_a^b f(x)\mathrm{d}x = f(c)(b - a) \tag{A.7}$$

图 A-4 中给出 (A.7) 的几何解释. 对于非负连续的 $f(x)$, (A.7) 的左边表示, 在 $x = a$ 与 $x = b$ 之间, 曲线 $y = f(x)$ 和 x 轴围成的面积. (A.7) 右边表示以 $[a, b]$ 为底、$f(c)$ 为高的长方形面积. 该定理断言, 在 a 与 b 之间存在某个 c, 使这两个面积相等.

结论 (A.6) 可用于一些积分的简化、估计以及求界. 这些情况大部分与 1.2 节中的泰勒定理有关. 作为积分中值定理应用的一个例子, 考虑积分

$$I = \int_a^b (b - x)(x - a)f(x)\mathrm{d}x$$

取 $w(x)=(b-x)(x-a)$. 则应用 (A.6) 知, 对 $[a,b]$ 中的某个 c,

$$I=f(c)\int_a^b(b-x)(x-a)\mathrm{d}x=\frac{(b-a)^3}{6}f(c) \tag{A.8}$$

这个积分出现在 5.2 节梯形数值积分法的分析中.

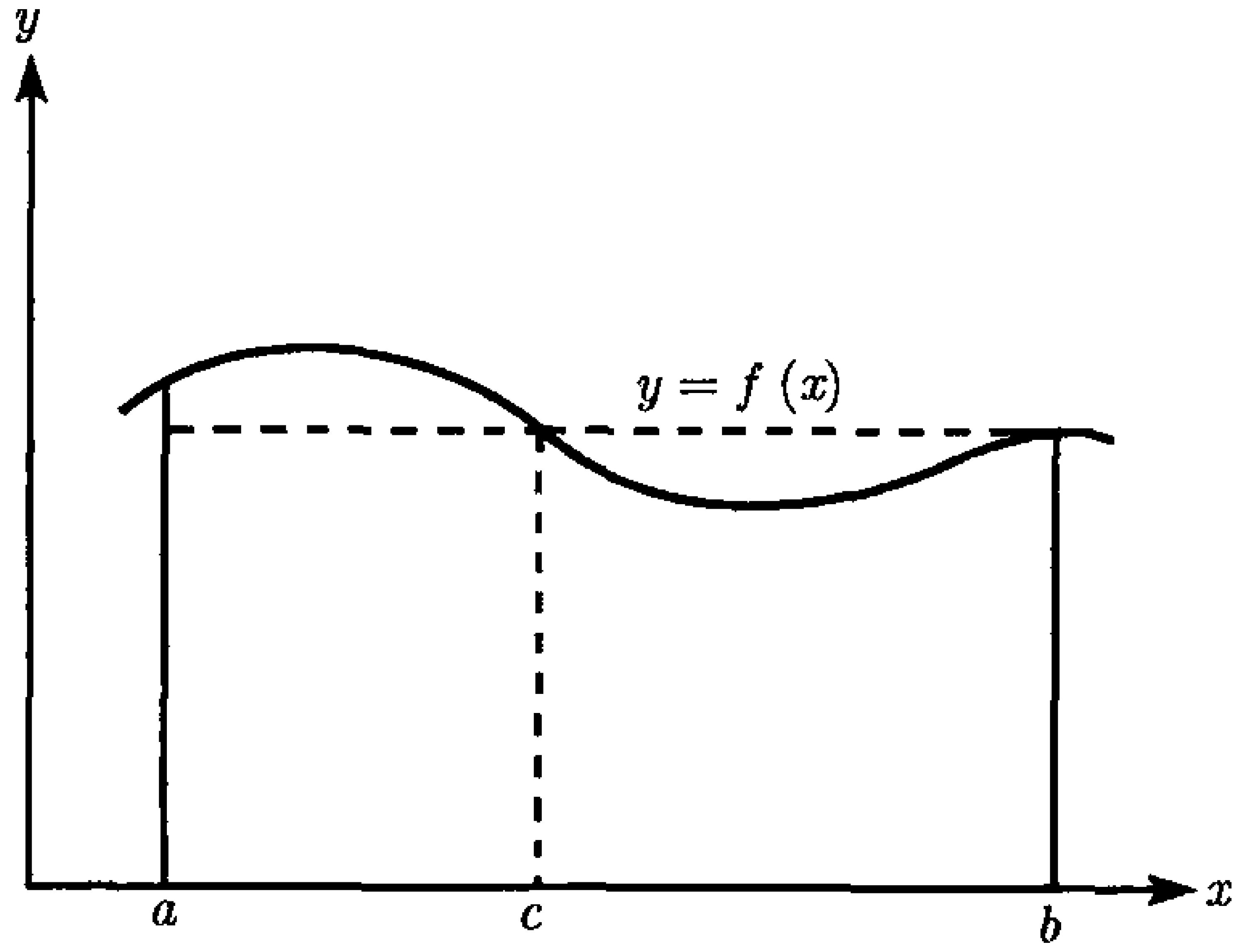


图 A-4 (A.7) 的说明

例 A.9 作为一个更具体的例子, 考虑估计

$$g(t)=\int_0^t\frac{x^n}{1+x^2}\mathrm{d}x,\quad n\geqslant 0$$

取 $w(x)=x^n$ 并且 $f(x)=1/(1+x^2)$. 则由 (A.6) 知, 对在 0 和 t 之间的某个 c, 有

$$g(t)=\frac{1}{1+c^2}\int_0^t x^n\mathrm{d}x=\left(\frac{1}{1+c^2}\right)\left(\frac{t^{n+1}}{n+1}\right)$$

因此对于 $t\geqslant 0$,

$$\left(\frac{1}{1+t^2}\right)\frac{t^{n+1}}{n+1}\leqslant g(t)\leqslant\frac{t^{n+1}}{n+1}$$ ■

习题

1. 通过求 c 的值使得对于给定的值 v 满足 $f(c)=v$ 来演示介值定理
   (a) $f(x)=2x^2-5x+2, v=-1, -5\leqslant x\leqslant 5$
   (b) $f(x)=\cos(x), v=1/2, \pi\leqslant x\leqslant 3\pi$
   (c) $f(x)=\sin(x)+\cos(x), v=\sqrt{2}, 0\leqslant x\leqslant \pi$
2. 如果对于 $a\leqslant x\leqslant b$, $f'(x)>0$, 这对 $f(x)=v$ 的解的个数有什么提示?
3. 仿照例 A.3, 利用 (A.2) 来求 $f(x)$ 在 $[a,b]$ 上的 "平均值". 并在 $[a,b]$ 内求实际得到 "平均值" 的点 c.
   (a) $f(x)=x^2, 0\leqslant x\leqslant 1$, $n=3$, 以及 $n=6$
   (b) $f(x)=\sin(x), 0\leqslant x\leqslant \pi, n=3$, 以及 $n=6$

4. 对下面选取的 $f(x)$ 和 $[a,b]$, 求满足 (A.3) 的 c 值, 从而演示中值定理.

(a) $f(x)=\cos(x), 0\leqslant x\leqslant \pi/2$

(b) $f(x)=x^3, 0\leqslant x\leqslant 1$

(c) $f(x)=x^3-x, -2\leqslant x\leqslant 2$

5. 估计用 $f(x_T)$ 代替 $f(x_A)$ 的误差.

(a) $f(x)=\sqrt{x}, x_T=\pi, x_A=22/7$

(b) $f(x)=\ln(x), x_T=\mathrm{e}, x_A=2.72$

(c) $f(x)=\cos(x), x_T=\sqrt{2}, x_A=1.414$

6. 推导下列不等式:

(a) 对所有 x,z, $|\cos(x)-\cos(z)|\leqslant|x-z|$

(b) 对所有 $x,z\leqslant 0$, $|\mathrm{e}^x-\mathrm{e}^z|\leqslant|x-z|$

(c) 对于 $-\pi/2<x,z<\pi/2$, $|\tan(x)-\tan(z)|\geqslant|x-z|$

(d) 对于 $0\leqslant y\leqslant x, p\geqslant 1$, $py^{p-1}(x-y)\leqslant x^p-y^p\leqslant px^{p-1}(x-y)$

7. 通过求使下列等式成立的点 c 来演示积分中值定理:

(a) $\displaystyle\int_0^1 x^3\mathrm{d}x=c^3$, c 在 $[0,1]$ 中

(b) $\displaystyle\int_0^1 \frac{x-(1/2)}{\sqrt{x}}\mathrm{d}x=\left(c-\frac{1}{2}\right)\int_0^1\frac{1}{\sqrt{x}}\mathrm{d}x=2\left(c-\frac{1}{2}\right)$, c 在 $[0,1]$ 内

(c) $\displaystyle\int_0^{\pi/2} x\sin(x)\mathrm{d}x=\sin(c)\int_0^{\pi/2}x\mathrm{d}x=\frac{\pi^2}{8}\sin(c)$, c 在 $[0,\pi/2]$ 内

8. 使用积分中值定理来推导下列结论:

(a) $\displaystyle\int_0^h x^2(h-x)^2g(x)\mathrm{d}x=\frac{h^5}{30}g(c)$, c 在 $[0,h]$ 内

(b) $\displaystyle\int_a^x (t-a)^n f(t)\mathrm{d}t=\frac{(x-a)^{n+1}}{n+1}f(c)$, c 在 $[a,x]$ 内

9. 我们可利用 (A.2) 得到 $f(x)$ 在 $[a,b]$ 上的平均值:

$$\text{平均值}=\frac{1}{n}\sum_{j=1}^{n}f(x_j)$$

这里, $x_1=a, x_2=a+h, x_3=a+2h,\cdots,x_n=b$, $h=(b-a)/(n-1)$. 如果将平均值写成

$$\frac{1}{b-a}\sum_{j=1}^{n}f(x_j)\left(\frac{b-a}{n}\right)$$

这个和式可看作积分的黎曼和. 让 $n\to\infty$, 则这个和式是 $f(x)$ 在 $[a,b]$ 上越来越精确的平均值, 它的极限等于

$$A=\frac{1}{b-a}\int_a^b f(x)\mathrm{d}x$$

如果使用 (A.7), 则对 $[a,b]$ 内的某个 c, 此式也可以写成 $A=f(c)$. 对于 A 使用积分公式计算习题 3 中函数的平均值并将结果与前面习题 3 的结论作比较.

附录B　数 学 公 式

B.1　代　　数

(a) 二次方程求根公式. $ax^2+bx+c=0(a\neq 0)$ 的解为

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}=\frac{2c}{-b\mp\sqrt{b^2-4ac}}$$

(b) 对数与指数. 对于 $a>0$,

$$a^x=\mathrm{e}^{x\ln a},\quad \log_a x=\frac{\ln x}{\ln a}$$

其中, $\ln x$ 表示以 e 为底的 x 的对数.

(c) 阶乘. $0!=1$

$$n!=n\cdot[(n-1)!]=n(n-1)(n-2)\cdots(2)(1),\quad n\geqslant 1$$

(d) 求和

$$\sum_{j=1}^{n} j=1+2+3+\cdots+n=\frac{n(n+1)}{2}$$

$$\sum_{j=1}^{n} j^2=1+4+9+\cdots+n^2=\frac{n(n+1)(2n+1)}{6}$$

$$\sum_{j=0}^{n} r^j=1+r+r^2+\cdots+r^n=\frac{r^{n+1}-1}{r-1},\quad r\neq 1$$

$$(a+b)^n=\sum_{j=0}^{n}\binom{n}{j}a^{n-j}b^j=\binom{n}{0}a^n+\binom{n}{1}a^{n-1}b+\cdots+\binom{n}{n}b^n$$

$$\binom{n}{j}=\frac{n!}{j!(n-j)!}$$

(e) 代数基本定理. 如果 $p(x)$ 是一个 n 次多项式, 则存在一组数 $r_1,r_2,\cdots,r_n$(可能为复数), 使得

$$p(x)=c(x-r_1)(x-r_2)\cdots(x-r_n)$$

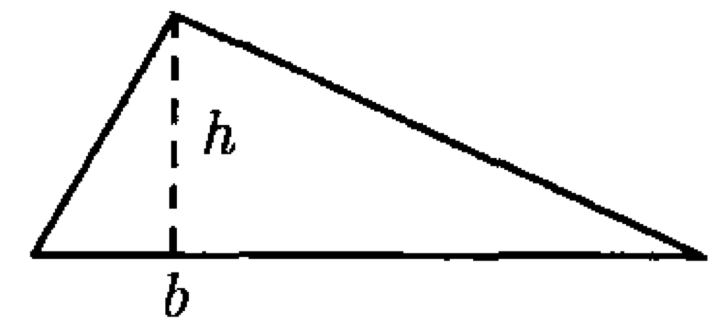
其中 c 是 $p(x)$ 中 x^n 的系数. 若不计数 $\{r_j\}$ 的顺序, 则它们是唯一的.

B.2 几　何

(a) 面积

(i) 三角形面积

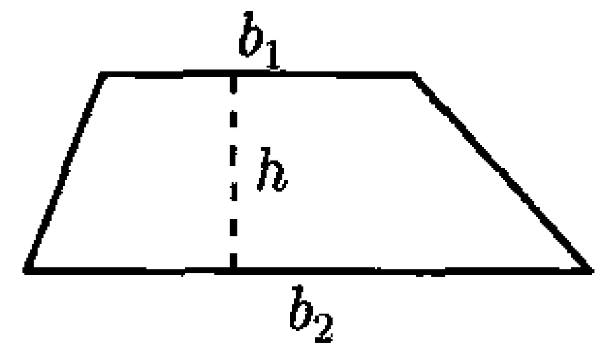
$$A = \frac{1}{2}bh \quad (b = 底, h = 高)$$



三角形

(ii) 梯形面积

$$A = h\left(\frac{b_1 + b_2}{2}\right)$$

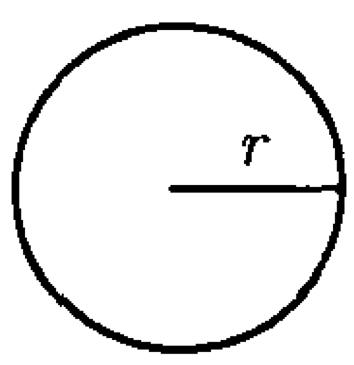


梯形

b_1 和 b_2 是平行边的长度.

(iii) 圆面积

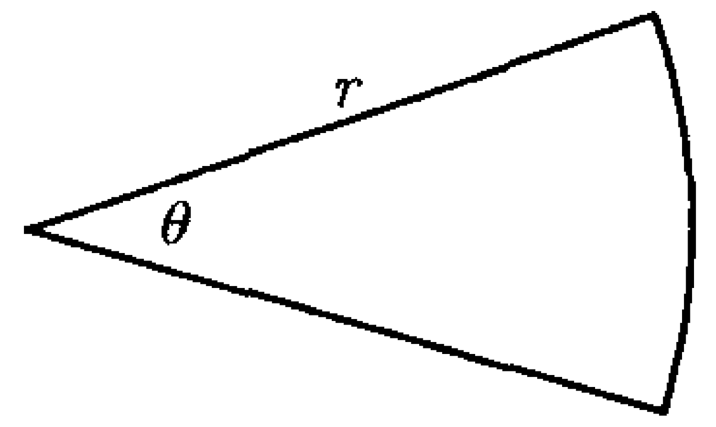
$A = \pi r^2 (r = 半径)$



圆

(iv) 中心角为 θ 弧度的圆的扇形面积

$$A = \frac{1}{2}r^2\theta$$



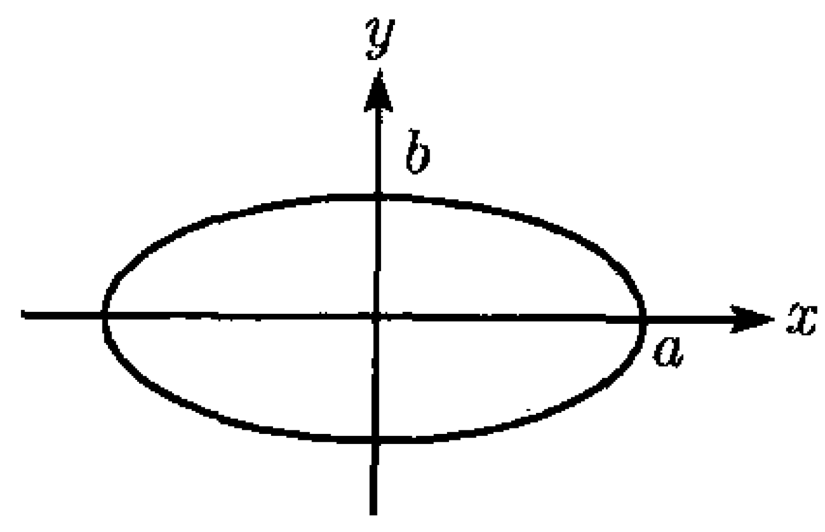
扇形

(v) 方程为

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

的椭圆面积

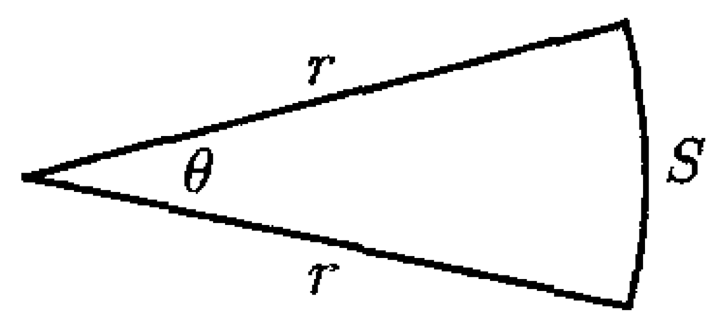
$$A = \pi ab$$



椭圆

(vi) 中心角为 θ 弧度的圆弧长

$$S = r\theta$$



圆弧

(b) 方程

(i) xy 平面内直线的方程

$$ax + by = c$$

(ii) 直线的斜率

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

(x_1, y_1) 和 (x_2, y_2) 是直线上的两个不同点, $x_1 \neq x_2$.

(iii) 三维空间中的平面方程

$$ax + by + cz = d$$

(iv) xy 平面内圆的方程

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

(x_0, y_0) 为圆心, r 为半径.

(v) 对称轴平行于 y 轴的抛物线方程

$$y = ax^2 + bx + c, \quad a \neq 0$$

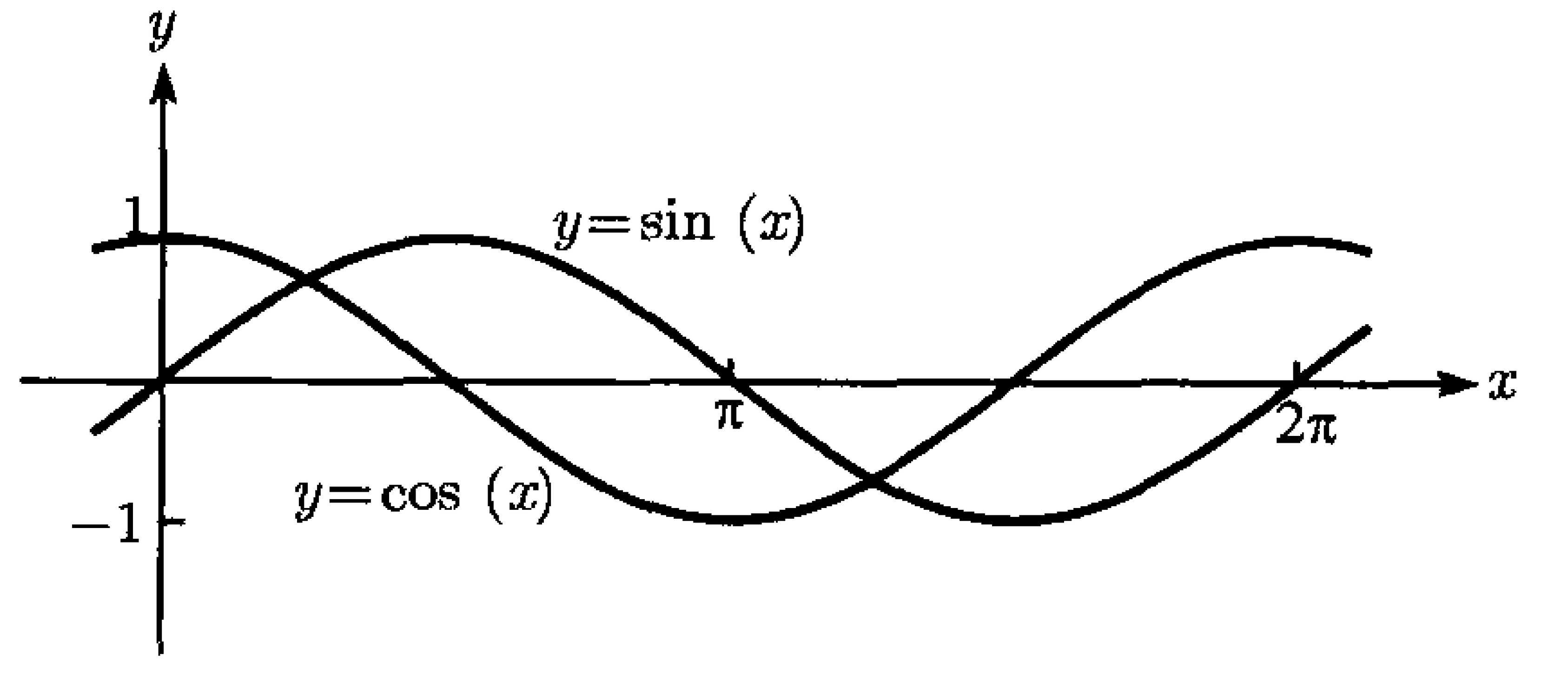
B.3 三 角

(a) 角的度数 θ_D 与角的弧度数 θ_R 之间的转换

$$\theta_R = \frac{\pi}{180}\theta_D$$

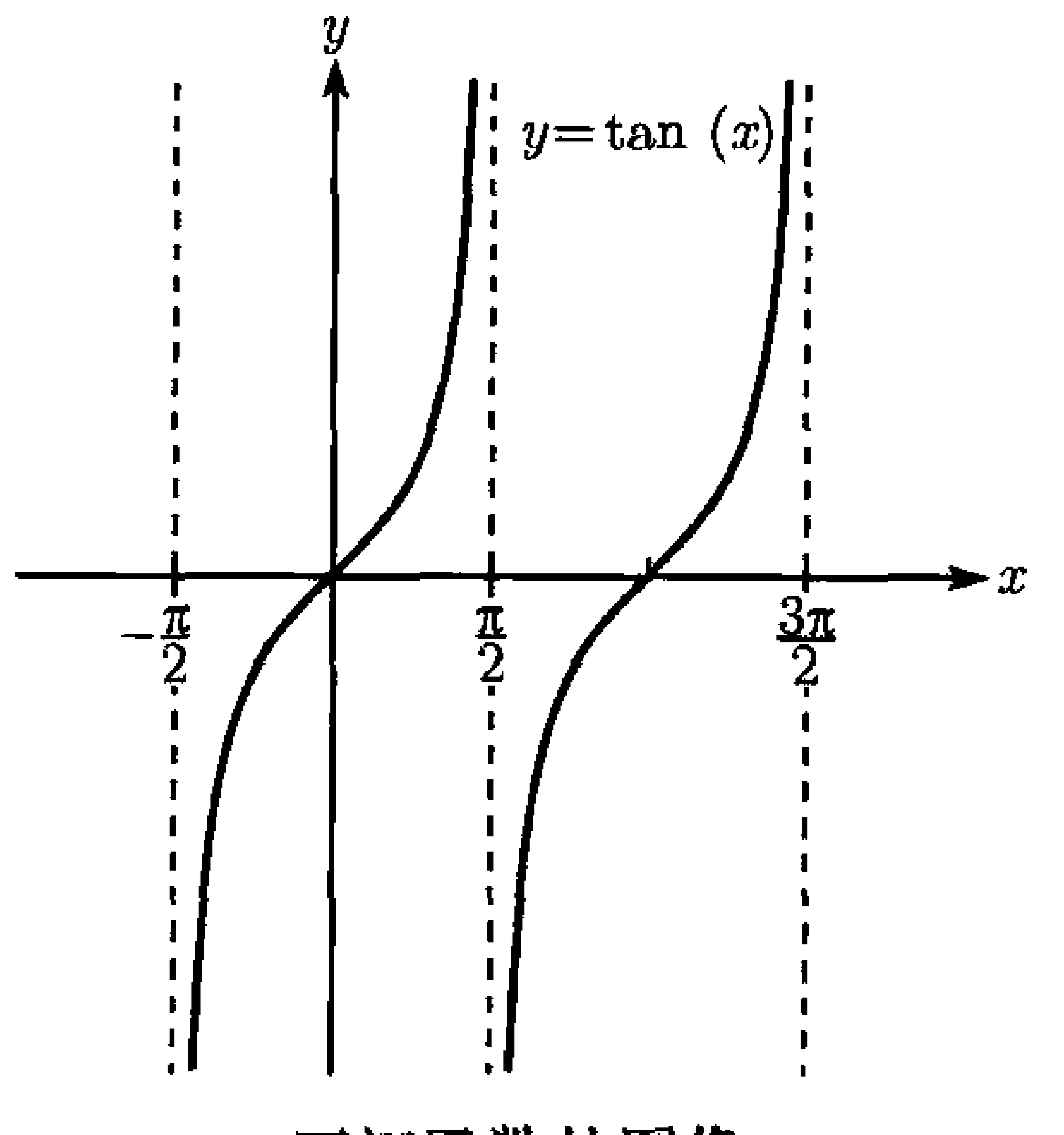
(b) 图形

(i) $\sin(x)$ 和 $\cos(x)$



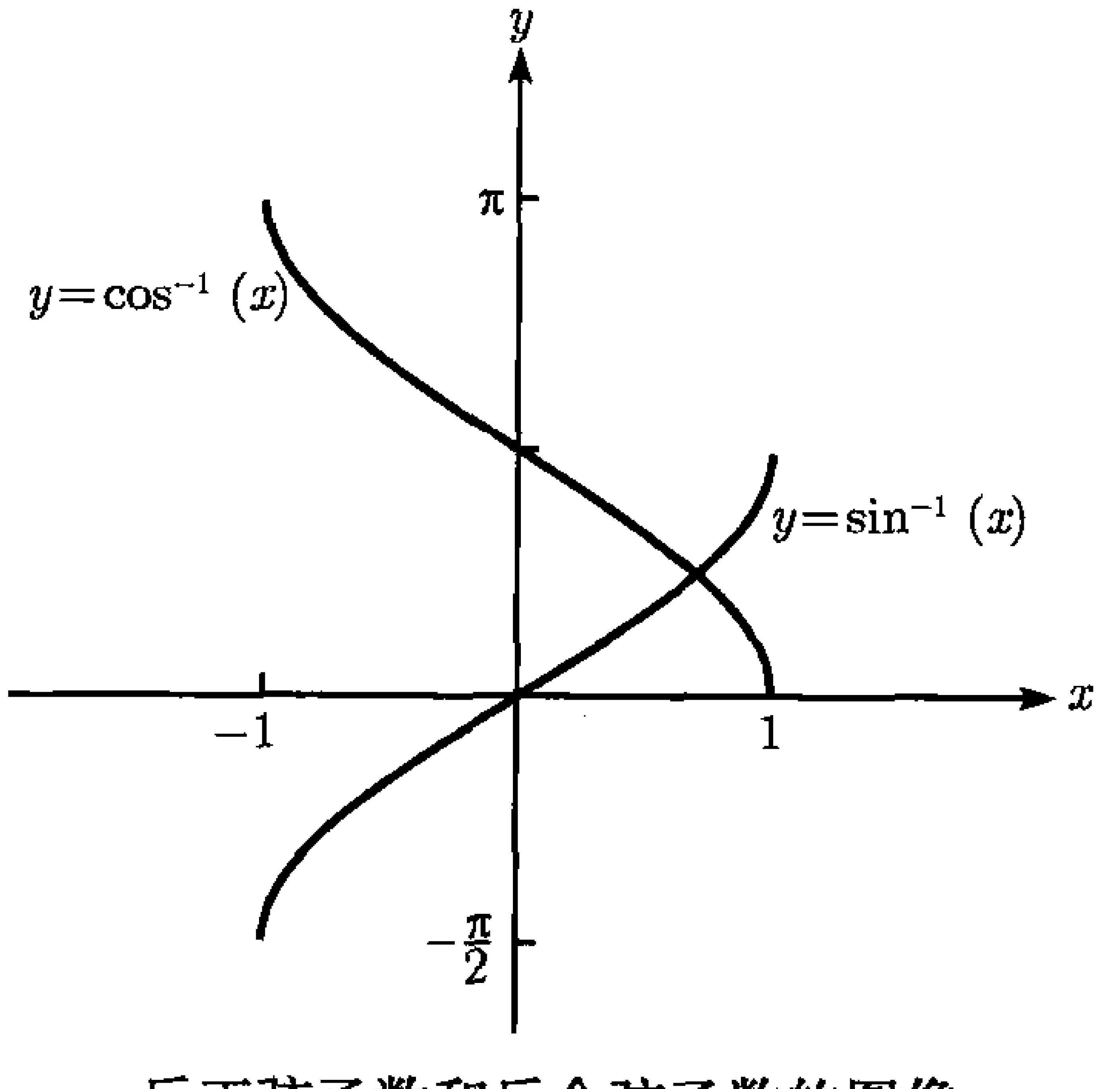
正弦函数和余弦函数的图像

(ii) $\tan(x)$



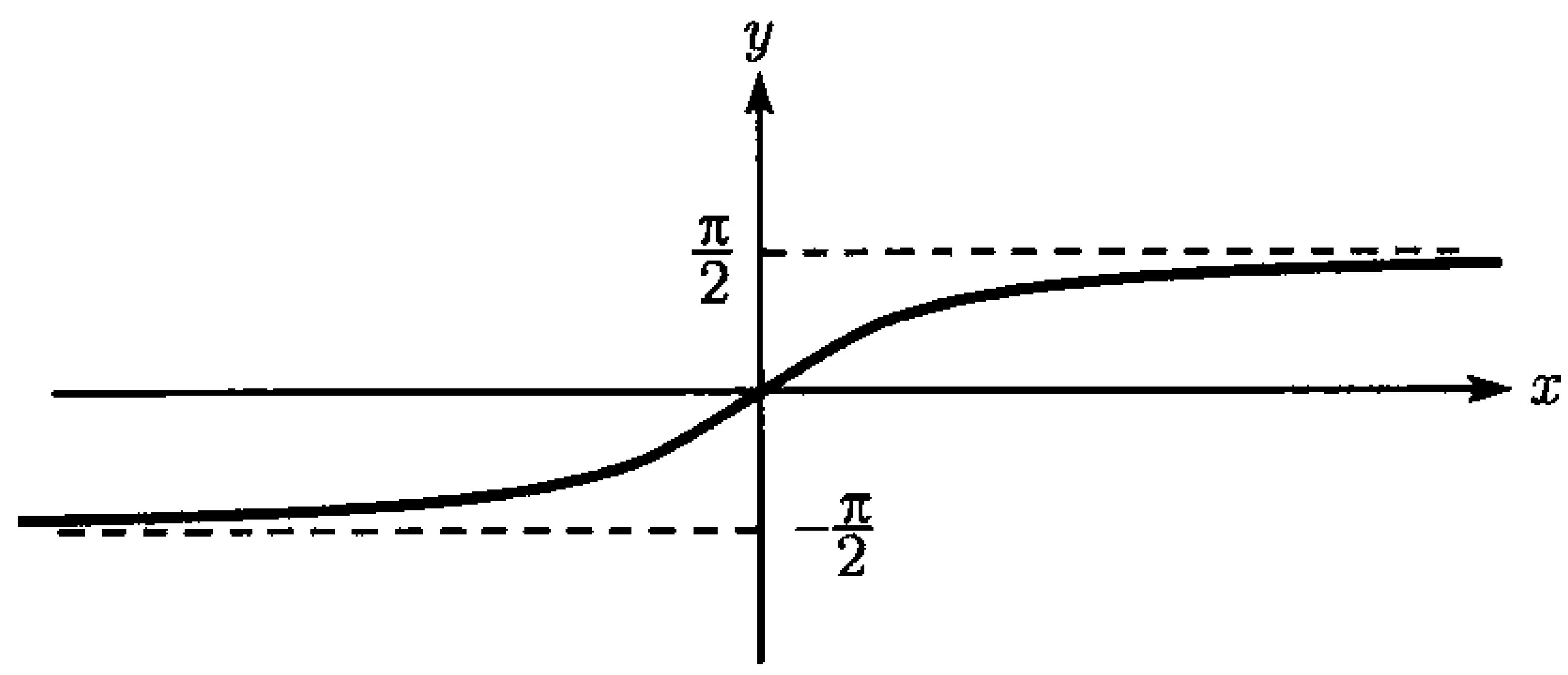
正切函数的图像

(iii) $\sin^{-1}(x) = \arcsin(x)$ 和 $\cos^{-1}(x) = \arccos(x)$



反正弦函数和反余弦函数的图像

(iv) $\tan^{-1}(x) = \arctan(x)$



反正切函数的图像

(c) 定义

$$\tan(x) = \frac{\sin(x)}{\cos(x)}, \quad \cot(x) = \frac{1}{\tan(x)} = \frac{\cos(x)}{\sin(x)}$$

$$\sec(x) = \frac{1}{\cos(x)}, \quad \csc(x) = \frac{1}{\sin(x)}$$

(d) 三角恒等式

$$\sin(-\alpha) = -\sin(\alpha), \quad \cos(-\alpha) = \cos(\alpha)$$

$$\cos^2(\alpha) + \sin^2(\alpha) = 1, \quad 1 + \tan^2(\alpha) = \sec^2(\alpha)$$

$$\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta)$$

$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$$

$$\tan(a \pm \beta) = \frac{\tan(\alpha) \pm \tan(\beta)}{1 \mp \tan(\alpha)\tan(\beta)}$$

$$\sin(2\alpha) = 2\sin(\alpha)\cos(\alpha)$$

$$\cos(2\alpha) = \cos^2(\alpha) - \sin^2(\alpha) = 2\cos^2(\alpha) - 1 = 1 - 2\sin^2(\alpha)$$

$$\sin^2(\alpha) = \frac{1 - \cos(2\alpha)}{2}, \quad \cos^2(\alpha) = \frac{1 + \cos(2\alpha)}{2}$$

$$\sin(\alpha) \pm \sin(\beta) = 2\sin\left(\frac{\alpha \pm \beta}{2}\right)\cos\left(\frac{\alpha \mp \beta}{2}\right)$$

$$\cos(\alpha) + \cos(\beta) = 2\cos\left(\frac{\alpha + \beta}{2}\right)\cos\left(\frac{\alpha - \beta}{2}\right)$$

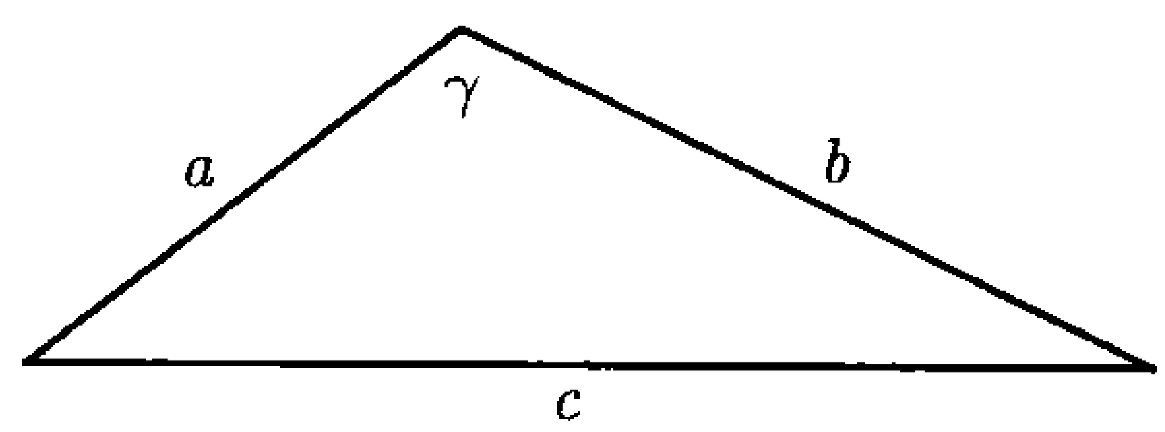
$$\cos(\alpha) - \cos(\beta) = -2\sin\left(\frac{\alpha + \beta}{2}\right)\sin\left(\frac{\alpha - \beta}{2}\right)$$

$$\sin^{-1}(x) + \cos^{-1}(x) = \frac{\pi}{2}$$

$$\tan^{-1}(x) = \frac{\pi}{2} - \tan^{-1}\left(\frac{1}{x}\right), \quad x > 0$$

(e) 余弦定律

$$c^2 = a^2 + b^2 - 2ab\cos(\gamma)$$



余弦定律

角 γ 的对边长度为 c.

B.4 微 积 分

(a) 导数的定义

$$f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

其他记号：$f'(x) = \dfrac{\mathrm{d}f(x)}{\mathrm{d}x} = \mathrm{D}f(x)$

(b) 导数的性质

$$[f(x)+g(x)]' = f'(x)+g'(x)$$

$$[f(x)g(x)]' = f(x)g'(x)+f'(x)g(x)$$

$$[cf(x)]' = cf'(x), \quad c = \text{常数}$$

$$[f(x)/g(x)]' = \frac{f'(x)g(x)-f(x)g'(x)}{[g(x)]^2}$$

$$[f(g(x))]' = f'(g(x))\cdot g'(x)(\text{链式法则})$$

(c) $\dfrac{\mathrm{d}}{\mathrm{d}x}(x^n) = nx^{n-1}, \quad n \neq 0$

$\dfrac{\mathrm{d}}{\mathrm{d}x}[\cos(x)] = -\sin(x), \quad \dfrac{\mathrm{d}}{\mathrm{d}x}[\sin(x)] = \cos(x)$

$\dfrac{\mathrm{d}}{\mathrm{d}x}[\tan(x)] = \sec^2(x), \quad \dfrac{\mathrm{d}}{\mathrm{d}x}[\sec(x)] = \sec(x)\tan(x)$

$\dfrac{\mathrm{d}}{\mathrm{d}x}[\sin^{-1}(x)] = \dfrac{1}{\sqrt{1-x^2}}, \quad \dfrac{\mathrm{d}}{\mathrm{d}x}[\cos^{-1}(x)] = \dfrac{-1}{\sqrt{1-x^2}}$

$\dfrac{\mathrm{d}}{\mathrm{d}x}[\tan^{-1}(x)] = \dfrac{1}{1+x^2}$

$\dfrac{\mathrm{d}}{\mathrm{d}x}[\mathrm{e}^x] = \mathrm{e}^x, \quad \dfrac{\mathrm{d}}{\mathrm{d}x}[a^x] = [\ln a]a^x$

$$\frac{\mathrm{d}}{\mathrm{d}x}[\ln(x)] = \frac{1}{x}, \quad \frac{\mathrm{d}}{\mathrm{d}x}[\log_a x] = \frac{1}{x \ln a}$$

(d) 定积分的定义

$$\int_a^b f(x)\mathrm{d}x = \lim_{n\to\infty} \sum_{j=1}^{n} f(p_j)(x_j - x_{j-1})$$

其中 $a = x_0 < x_1 < x_2 < \cdots < x_n = b, p_j$ 在 $[x_{j-1}, x_j]$ 中, 并且当 $n \to \infty$ 时, $\max(x_j - x_{j-1}) \to 0$. 上面的和式称为黎曼和.

(e) 定积分的性质

$$\int_a^b [f(x) + g(x)]\mathrm{d}x = \int_a^b f(x)\mathrm{d}x + \int_a^b g(x)\mathrm{d}x$$

$$\int_a^b cf(x)\mathrm{d}x = c\int_a^b f(x)\mathrm{d}x, \quad c = \text{常数}$$

$$\int_a^b f(x)\mathrm{d}x = \int_a^c f(x)\mathrm{d}x + \int_c^b f(x)\mathrm{d}x$$

$$\int_a^b f(x)\mathrm{d}x = -\int_b^a f(x)\mathrm{d}x$$

$$\int_a^b f(x)g'(x)\mathrm{d}x = [f(x)g(x)]_a^b - \int_a^b f'(x)g(x)\mathrm{d}x$$

$$\int_a^b f(g(x))g'(x)\mathrm{d}x = \int_{g(a)}^{g(b)} f(u)\mathrm{d}u$$

(f) 微积分基本定理

形式 1 若 $F(x)$ 是 $f(x)$ 在 $[a, b]$ 上的一个原函数, 则

$$\int_a^b f(x)\mathrm{d}x = [F(x)]_{x=a}^{x=b} = F(b) - F(a)$$

形式 2

$$\frac{\mathrm{d}}{\mathrm{d}x}\int_a^x f(t)\mathrm{d}t = f(x)$$

(g) 不定积分表

$$\int x^n \mathrm{d}x = \frac{x^{n+1}}{n+1} + c, \quad n \neq -1$$

$$\int \frac{\mathrm{d}x}{x} = \ln|x| + c$$

$$\int \mathrm{e}^x \mathrm{d}x = \mathrm{e}^x + c, \quad \int a^x \mathrm{d}x = \frac{a^x}{\ln(a)} + c$$

$$\int x^{\alpha}\ln(x)\mathrm{d}x = \frac{x^{\alpha+1}}{\alpha+1}\cdot\ln(x) - \frac{x^{\alpha+1}}{(\alpha+1)^2} + c, \quad \alpha \neq -1$$

$$\int \frac{\mathrm{d}x}{x^2+a^2} = \frac{1}{a}\tan^{-1}\left(\frac{x}{a}\right) + c$$

$$\int \frac{\mathrm{d}x}{\sqrt{a^2-x^2}} = \sin^{-1}\left(\frac{x}{a}\right) + c$$

$$\int \sin(x)\mathrm{d}x = -\cos(x) + c, \quad \int \cos(x)\mathrm{d}x = \sin(x) + c$$

$$\int \sec^2(x)\mathrm{d}x = \tan(x) + c, \quad \int \sec(x)\tan(x)\mathrm{d}x = \sec(x) + c$$

$$\int \tan(x)\mathrm{d}x = \ln|\sec(x)| + c$$

$$\int \sec(x)\mathrm{d}x = \ln|\sec(x) + \tan(x)| + c$$

$$\int \sin^2(x)\mathrm{d}x = \frac{x}{2} - \frac{\sin(2x)}{4} + c$$

$$\int \cos^2(x)\mathrm{d}x = \frac{x}{2} + \frac{\sin(2x)}{4} + c$$

附录C　数值分析软件包

本书中包含程序, 是为了帮助读者理解数值方法. 这些程序并不一定满足高精度、有效、稳定和便捷的要求. 在过去的三十多年里, 已经产生了许多高质量的数据分析软件包. 本附录列举其中若干个软件包, 以及得到这些软件包所需的相关信息.

Fortran 是编写数值计算程序的传统的程序设计语言. 目前许多高质量的软件包是用它编写的, 这也不足为奇. 近年来, C、C++ 以及 Java 也在计算中越来越流行. 许多被广泛使用的 Fortran 软件包同时也有 C 语言的版本. 另外, 用户可以使用贝尔实验室研制的 `f2c` 转换器方便地将 Fortran 原始代码转换为 C 语言. `f2c` 转换器的免费下载地址是 `http://netlib.bell-labs.com/netlib/f2c`.

C.1　商用软件包

目前, 在数值分析领域, 有两种使用广泛的通用数值计算软件包, 每一种都含有可以求解大多数标准数值分析问题的程序. 大多数计算机上都可以装配这两种软件包.

1. IMSL(International Mathematics and Statistics Library)　由位于德克萨斯州休斯敦 (Houston) 的可视数字化股份有限公司出品, 网址是 `www.vni.com/products/imsl`. 含有三个 IMSL 数据库: IMSL Fortran 90MP 数据库、IMSL C 语言数值数据库以及用于 Java 应用开发的 JMSL 数据库. IMSL 是一个集数学函数与统计学函数于一体的数据库, 在为科学计算进行编程时, IMSL 可以作为可靠的编写平台.

2. NAG(Numerical Algorithms Group)　由位于伊利诺伊州丹尼格罗夫 (Downers Grove) 的 NAG 股份有限公司出品, 网址是 `www.nag.com`. NAG 含有用以解决诸多应用问题的数值程序、符号程序、统计学程序、可视化程序以及模拟程序. 程序大部分用 Fortran 或 C 语言编写.

C.2　共享软件包

尽管部分软件包被作为商品出售, 但是还有些软件包可以通过互联网免费共享.

获取免费软件的一个有效途径是通过 Netlib, 它汇集了大量的数学软件. Netlib 中所有资源的制作和维护是由贝尔实验室、田纳西大学和橡树山国家实验室共同负责的, 可以免费利用其中所有的程序. 搜索 Netlib 相关软件或文件, 可以输入网址

`www.netlib.org`

或者发电子邮件 (将信息 `send index` 送至 `netlib@netlib.org`). 也可以通过 FAQ(常用问题解答网) 来初步了解 Netlib, FAQ 网址为

`http://www.netlib.org/misc/faq.html#2.1`

这里我们列举若干 Netlib 中的软件包:

1. APPACK APPACK 是一个 Fortran77 的子程序集合, 用来计算一般的大型矩阵的几个本征值和对应的本征向量. 它的并行算法的版本 PARPACK 也可以在网上找到. 主要参考文献是:

R. B. Lehoucq, D. C. Sorensen, and C. Yang, A*RPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia,1998.

2. BLAS(Basic linear Algebra Subprograms) BLAS 是矩阵和向量计算的有效、便捷、高质量的 "内置块" 子程序. BLAS 分为三个级别, 分别是向量–向量运算、矩阵–向量运算、矩阵–矩阵运算. BLAS 广泛应用于研发高质量的数值线性代数软件, 例如 LINPACK 和 LAPACK. Fortran 实施的 BLAS、C 语言界面的 BLAS 和 Java 语言的 BLAS 都可以在网上找到. 相关的参考文献是:

C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, Basic Linear Algebra Subprograms for FORTRAN Usage, *ACM Trans. Math. Soft.* **5** (1979), 308-323.

J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.* **14** (1988).1-17.

J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, Algorithm 656: An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.* **14** (1988), 18-32.

J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.* **16** (1990), 1-17.

J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, Algoruthm 679: A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.* **16** (1990), 18-28.

3. EISPACK 该软件包采用双精度 Fortran, 用于求解各种不同类型矩阵的本征值及本征向量问题. 单精度版本的子程序在 SEISPACK 中给出. 与其他一些软

件包一样, 它是由阿尔贡 (Argonne) 国家实验室和其他一些大学的研究人员合作研发的, 美国国家自然科学基金出资赞助. 要得到更加详细描述的软件包和其中所有程序的详尽清单, 可参见下列专题文章:

B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines-EISPACK Guide*, 2nd ed., *Lecture Notes in Computer Science*, Volume 6, Springer-Verlag, New York, 1976.

B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B.Moler, *Matrix Eigensystem Routines- EISPACK Guide Extention, Lecture Notes in Computer Science,* Volume 51, Springer-Verlag, New York, 1977.

4. FFTPACK 这是一个用于周期序列和其他对称序列的快速傅里叶变换的 Fortran 子程序的软件包. 主要参考文献是

P. N. Swarztrauber, Vectorizing the FFTs, in *Parallel Computations*, G. Rodrigue, ed., Academic Press, New York, 1982, pp. 51-83.

5. FISHPACK 这个软件包含求解椭圆边值问题有限差分方法的 Fortran 程序.

6. LAPACK 这个软件包组合、更新和扩展了 EISPACK 和 LINPACK 中的程序, 可用于求解线性代数方程组、线性最小二乘解、本征值问题以及奇异值问题. 程序的版本适合四种计算机算术运算: 单精度和双精度实数运算以及单精度和双精度复数运算. 特别地, 这些程序能很好地适应新一代的向量计算机和并行计算机. 这个软件包完整的叙述参见下列书籍:

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Vrot, A. Greenhaum S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensent, LAPACK Users' Guide, third edition, SIAM, Philadephia, 1999.

LAPACK 程序用 Fortran77 编写. 它的 C 语言版本称为 CLAPACK, C++ 版本是 LAPACK++, Fortran99 界面是 LAPACK90.

7. LINPACK 这个软件包用于求解线性代数方程组. 其中包含一些特殊的子程序, 用于许多具有相同类型系数矩阵的方程组. 具备有效的误差控制, 并能检验线性方程组系数矩阵的条件. LINPACK 在很大程度上已被 LAPACK 取代. 有关 LINPACK 中的文件和所有程序的目录, 参见

J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK Users' Guide*, SIAM Publications, 1979.

8. MINPACK 它包括用于求解非线性代数方程组和非线性最小二乘问题的程序. 阿尔贡 (Argonne) 国家实验室开发了这个软件.

9. ODEPACK 它汇集了求解常微分方程组初值问题的 Fortran77 的程序. 可用来求解刚性方程组和非刚性方程组. 更多信息参见

A. C. Hindmarsh, ODEPACK, A Systematized Collection of ODE Solvers, in *Scientific Computing*, R. S. Stepleman et al., eds., North-Holland, Amsterdam, 1983 (Vol. 1 of *IMACS Transactions on Scientific Computation*), pp. 55-64.

10. PLTMG 这个软件包包含求解一般平面区域中的椭圆型偏微分方程的 Fortran 程序. 它适合于局部网格加细、多重网格迭代和依赖于参数的拟弧长延拓. 主要参考文献是

R. Bank, PLTMG: *A software Package for solving Elliptic Partial Differential Equations—Users' Guide 8.0*, SIAM, Philadelphia, 1998.

11. QUADPACK 这是一维定积分数值计算的 Fortran 子程序包. 大部分程序都是自动的, 并且尝试在绝对误差或者相对误差的允许范围内完成积分. 更多信息请参见

R. Piessens, E. deDoncker-Kapenga, C. Uberhuber, and D. Kahaner, *Quadpack: A Subroutine Package for Automatic Integration*, Springer Verlag, New York, 1983.

数学和统计学软件的另一个好的来源是, 从 NIST(National Institute of Standards and Technology) 得到数学软件指南, 网址是 `gams.nist.gov`. 除了能帮助找到 Netlib 中的软件包之外, 这个网站还包含很多其他的软件包.

C.3 交互的数值计算环境

在科研和教学工作中, 交互的计算环境越来越流行. 这种交互的环境易于学习, 使用方便, 具备强大的内置的数学功能. 利用这种交互环境, 我们可以把更多的精力集中在问题的建模和求解以及结论的表述, 而不必过多地关注程序细节.

一个广泛使用的交互计算环境是 MATLAB. 它是位于马塞诸塞州纳提克 (Natick) 的 Mathworks 股份有限公司的商品, 网址是 `www.mathworks.com`. 本书使用这个交互计算环境. 附录 D 中简单介绍了 MATLAB, 关于它的更多细节可参见附录 D 中的参考文献.

在市场中, 有许多像 MATLAB 一样的计算环境. 它们都提供了以矩阵为基本数据类型的可以解释数学程序的语言和简单的句法. 例如:

- *GAUSS* of APTECK Systems, Inc., `www.apteck.com`
- *MathViews* of MathWizards, `www.mathwizards.com`
- *Mathcad* of Mathsoft Engineering & Education Inc., `www.mathsoft.com`
- *O*−MATRIX, `www.omatrix.com`
- *PV-WAVE* of Visual Numerics, Inc., `www.vni.com/products/wave`

也有一些免费提供的像 MATLAB 一样的计算环境. 例如:

- *Octave* by John W. Eaton and others, `www.octave.com`

- *RLaB*, which is no longer under active development, `rlab.sourceforge.net`
- *Scilab* by the Scilab Group at INRIA of France, `www-rocq.inria.fr/scilab`
- *Tela* (TEnsor LAnguage), `www.geo.fmi.fi/prog/tela.html`

C.4 符号计算环境

80 年代, 迎来了用计算机程序做符号数学而不是数值数学的时代. 越来越多地使用符号数学的程序, 改变了包括数值分析在内的数学的教学和科研工作. 使用这些新的计算机工具将把我们带向何方尚不清楚, 但是它们可能会对数学的所有领域的实践产生重大影响. 下面列举一些主要的计算机代数系统:

- *Derive* of MathWare Ltd., `www.mathware.com`
- *Macsyma*, `www.scientek.com/macsyma/mxmain.htm`
- *Maple* of the Symbolic Computation Group, University of Waterloo, `www.scg.uwaterloo.ca`
- *Mathematica* of Wolfram Research Inc., `www.wolfram.com`
- *MuPAD*, `www.mupad.de`
- *Reduce*, `www.uni-koeln.de/REDUCE`

MATLAB 也有一个符号数学工具箱, 它与 Maple 引擎的链接可以实现多种符号计算.

C.5 数学软件的文献

有一些期刊刊登有关数学软件的新程序或者信息. 其中几种期刊由计算机协会 (ACM) 出版, 网址是 `www.acm.org`, 我们把参考文献限定在这些期刊中.

1. *ACM Transactions on Mathematical software.* 这个杂志包含关于数值分析软件的开发和检测的文章. 它也有各种各样的数值分析问题的一些新的程序. 这些程序通常非常仔细地编写, 可以在各种不同的计算机上使用.

2. *Collected Algorithms of the ACM.* 这份杂志集中了所有已发表在 ACM 杂志上的程序, 这个程序汇集可以单独向 ACM 订购. 现在所有新的数学程序都发表在 *ACM Transactions software* 中. 算法的机器可读副本可从 NETLIB 得到.

这个附录里的讨论远远没有包括所有高质量的数字软件. 另外, 附录中倾向于作者们熟悉的程序.

附录D MATLAB 简介

MATLAB 是为实现数值分析中需要的各种数值计算而特别设计的一种交互式计算机语言. 在 MATLAB 中可如同在计算器上一样直接给出指令, 并且可编写程序. 学习 MATLAB 的最佳方法是使用它. 许多 MATLAB 表达式和指令有一种简单使用的默认 (default) 格式, 这些格式可容易地扩充许多选项. 本附录可帮助读者初步理解 MATLAB. 已有许多关于 MATLAB 的著作和在线指南, 本附录的末尾给出一些相关的材料.

如同大多数计算机语言那样, 标准的数学表达式可以求值. 例如, 表达式

$$y = 6 - 4x + 7x^2 - 3x^5 + \frac{3}{x+2} \tag{D.1}$$

根据前面给定的 x 值, 利用

$$\mathsf{y = 6 - 4 * x + 7 * x * x - 3 * x\hat{~}5 + 3/(x + 2);} \tag{D.2}$$

求值. 注意, 语句的末尾用一个分号. 这在 MATLAB 语句中经常用到. 省略分号会导致在计算机屏幕上打印 y 值. MATLAB 有许多内置函数, 例如 e^x、$\ln x$、三角函数和反三角函数、$\sqrt{x}$ 和其他一些标准初等函数. 也有更复杂的函数, 例如 $\mathrm{gamma}(x)$ 是在 x 处赋值的标准的伽玛函数; 对非负整数 n, 阶乘函数 $\mathrm{gamma}(n+1) = n!$. 重要的常数 π 利用变量名 `pi` 可以得到.

MATLAB 中默认使用双精度和实的算术运算, 而在适当的地方默认复数算术运算. 例如, `sqrt(-4)`产生答案 `2i`. 输出到屏幕的数据默认值在十进制小数点的右边有 4 位数字. 控制数据屏幕输出的格式使用命令 `format`. 利用

```
format short
```

得到默认值. 为得到全精度的数, 你可以使用命令

```
format long
```

下面两个命令

```
format short e
format long e
```

将给出数据的 “科学记数法” 的输出. 还可以利用其他的 `format` 选项.

MATLAB 的一个重要功能是它对数组以及许多用数组操作的任务十分有效. 我们以画出区间 $0 \leqslant x \leqslant 10$ 上 $\sin x$ 和 $\cos x$ 图形的简单程序来说明这点.

$$\begin{aligned}&\mathtt{t = 0 : .1 : 10;}\\&\mathtt{x = cos(t); y = sin(t);}\\&\mathtt{plot(t, x, t, y)}\end{aligned} \tag{D.3}$$

第一行建立一个长度为 101 的行向量:

$$\boldsymbol{t} = [0, 0.1, 0.2, \cdots, 1.0, 1.1, \cdots, 9.9, 10]$$

一般说来, 语句

$$\mathtt{t = a : h : b;}$$

产生下列形式的行向量

$$\boldsymbol{t} = [a, a+h, a+2h, \cdots]$$

其中的分量为所有 $\leqslant b$ 的形如 $a + jh$ 的值. 如果 $h = 1$, 可省略 h 不写. 如

$$\mathtt{n = 1 : 5}$$

产生一个行向量

$$\boldsymbol{n} = [1, 2, 3, 4, 5]$$

(D.3) 中程序的第二行定义了两个新的行向量. 指令 `x = cos (t)` 产生一个与 `t` 一样长的行向量, 而 `x` 由 `t` 的所有分量处的余弦函数值组成. 指令 `y = sin (t)` 的操作是类似的. (D.3) 中的第三行根据 `t`, `x`, `y` 中存放的值, 同时画出区间 $[0, 10]$ 上余弦函数和正弦函数的图形. 读者应该试做这个简单的例子 (以及本附录中给出的其他一些例子). 此例也说明, 在 MATLAB 中产生图形的简单性.

在 MATLAB 中, 可以用多种方式产生数组. 例如

$$\mathtt{b = [1, 2, 3]}$$

产生一个长度为 3 的行向量; 而

$$\mathtt{A = [1\ 2\ 3;\ 4\ 5\ 6;\ 7\ 8\ 9]}$$

产生一个方阵

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

可用空格和逗号作为所给数组分量的间隔; 而分号分隔矩阵的不同的行, 如上例所示. 当要定义列向量时, 可以使用转置操作. 例如

$$\texttt{b} = [1\ 3\ \ -6]'$$

产生列向量

$$\begin{bmatrix} 1 \\ 3 \\ -6 \end{bmatrix}$$

当两个数组有同样大小时, 按标准的矩阵方式相加它们, 即按分量相加. 例如, 相加 2×3 阶矩阵

$$\begin{aligned} &\texttt{A} = [1, 2; 3, -2; -6, 1]; \\ &\texttt{B} = [2, 3; -3, 2; 2, -2]; \\ &\texttt{C} = \texttt{A} + \texttt{B}; \end{aligned}$$

答案为

$$\boldsymbol{C} = \begin{bmatrix} 3 & 5 \\ 0 & 0 \\ -4 & -1 \end{bmatrix}$$

当用常量乘以一个数组时, 我们用常量乘以数组的每个分量. 例如, 利用上面的矩阵 A 和指令

$$\texttt{D} = 2 * \texttt{A};$$

答案为

$$\boldsymbol{D} = \begin{bmatrix} 2 & 4 \\ 6 & -4 \\ -12 & 2 \end{bmatrix}$$

矩阵乘法具有同样的标准意义. 例如, 下列矩阵相乘:

$$\begin{aligned} &\texttt{E} = [1, -2; 2, -1; -3, 2]; \\ &\texttt{F} = [2, -1, 3; -1, 2, 3]; \\ &\texttt{G} = \texttt{E} * \texttt{F}; \end{aligned}$$

答案为

$$\boldsymbol{G} = \begin{bmatrix} 1 & -2 \\ 2 & -1 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 & 3 \\ -1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 4 & -5 & -3 \\ 5 & -4 & 3 \\ -8 & 7 & -3 \end{bmatrix}$$

一个非标准记号是常量和数组的加法. 把常数与数组的每个分量相加来实施这个加法. 例如, 利用上面的矩阵 F, 指令

$$\mathtt{H = 3 + F;}$$

计算的结果为

$$\boldsymbol{H} = 3\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 2 & -1 & 3 \\ -1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 6 \\ 2 & 5 & 6 \end{bmatrix}$$

6.1 节和 6.2 节详细地讨论了这些运算及其实施方法.

MATLAB 也有按分量的乘法、除法和指数运算, 这些运算在大部分线性代数课程中没有使用过. 在通常的运算符号前面加上一个小数点来表示这三种运算. 例如, 对

$$\mathsf{a} = [1 \quad 2 \quad 3]; \quad \mathsf{b} = [2 \quad -1 \quad 4];$$

我们有

$$\begin{aligned} \mathsf{a.*b} &= [\,2 \quad -2 \quad 12\,] \\ \mathsf{a./b} &= \left[\,0.5 \quad -2.0 \quad 0.75\,\right] \\ \mathsf{a.\hat{}3} &= [\,1 \quad 8 \quad 27\,] \\ \mathsf{2.\hat{}a} &= [\,2 \quad 4 \quad 8\,] \\ \mathsf{b.\hat{}a} &= [\,2 \quad 1 \quad 64\,] \end{aligned}$$

作为 MATLAB 中按分量运算更进一步的例子, 考虑在数组 x 的所有分量处求 (D.1) 中代数表达式的值. 因而 MATLAB 指令 (D.2) 应改写为

$$\mathsf{y = 6 - 4*x + 7*x.*x - 3*x.\hat{}5 + 3./(x+2);}$$

注意, .*, .∧ 和 ./ 使用按分量运算. 其余的运算具有以前线性代数讨论过的标准意义.

正如 (D.3) 的程序中看到的那样, 我们也可以对数组应用标准函数, 把函数理解为按分量进行计算的函数. 例如,

$$\mathsf{sqrt}\,([1\ 4\ 9]) = [1\ 2\ 3]$$

有许多涉及 “事务管理” 细节的 MATLAB 运算. 例如, 利用指令 `who` 可知道当前有效的所有变量. 利用指令 `whos` 可进一步得到这些变量的大小和类型. 利用指令 `clear` 可去掉当前使用的变量. 利用指令 `cls` 清除屏幕输出. 利用格式

```
help command_name
```

得到 MATLAB 指令的描述. 例如,

```
help sqrt
```

输出结果为

```
SQRT Square root.
SQRT(X)is the square root of the elements of X. Complex
results are produced if X is not positive.
```

还有一些产生特殊矩阵的指令. 一个常用的指令如下:

$$\mathtt{A} = \mathtt{zeros}(2,3)$$

产生一个所有分量为零的两行三列的数组

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

这个函数类似于 *Fortran* 中数组初始化为零的 DIMENSION 语句. 指令

$$\mathtt{B} = \mathtt{ones}(2,3)$$

产生一个所有分量为 1 的两行三列数组

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

这个矩阵不同于利用指令 `eye` 产生的恒等矩阵. 例如, 指令 `eye(3)` 产生 3×3 恒等矩阵

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

有许多对数组运算的 MATLAB 指令, 这里我们只给出其中几个. 对长度为 n 的行向量或列向量 x, 有下列函数:

$\mathtt{max}(\mathtt{x}) = \mathtt{x}$ 的最大分量
$\mathtt{min}(\mathtt{x}) = \mathtt{x}$ 的最小分量
$\mathtt{abs}(\mathtt{x}) = \mathtt{x}$ 的分量绝对值构成的向量
$\mathtt{sum}(\mathtt{x}) = \mathtt{x}$ 的分量之和

$$\mathtt{norm}(\mathtt{x}) = \sqrt{|x_1|^2 + \cdots + |x_n|^2}$$

利用 `help` 指令可以得到有关这些函数更多的信息.

脚本式文件和函数式文件 虽然 MATLAB 是一种交互式语言, 可以从键盘输入指令并逐次执行, 但是通常有必要创建一个你要 MATLAB 执行的指令文件. 例如, 可以把 (D.3) 中的程序作为一个文件存放起来, 然后执行. 这种文件必须有一个以. m (小数点后接字母 m) 结尾的名称. 例如, 可以在文件名 demo_plot.m 之下存放 (D.3), 然后可以简单地从键盘输入 demo_plot 并回车执行它. 这样的一个文件成为脚本式 (script) 文件. 这种文件有许多优点. 当给出交互式命令时很容易输入出错, 使用脚本式文件时候比较容易纠正这种误差. 我们也可编写一个程序进行实验, 随需要进行调整. 例如, 我们可对 demo_plot.m 增加一些指令从而对脚本产生的图形增加标号和标题. 更重要的是我们可以编写较复杂的程序. 例如 1.1 节中给出的名称为 exp_taylor 的程序, 它对 e^x 的各种泰勒多项式逼近产生一个近似值表; 本书的其他地方还有别的例子.

也可以编写 MATLAB 的函数文件, 这样可以产生许多新的函数. 函数有输入和输出参数; 在函数中使用的其他变量在函数外失效. 例如, 考虑求多项式

$$p(x) = a_1 + a_2x + a_3x^2 + \cdots + a_nx^{n-1}$$

的值的函数. 因为 MATLAB 不允许数组下标为零, 所以开始的指标用 1 而不是 0. 下列函数应该存放在名称 polyeval.m 之下. 函数在名称为 coeff 的数组中给出系数 $\{a_j\}$, 而多项式在数组 x 的所有分量处求值.

```
function value = polyeval(x,coeff);
%
% function value = polyeval(x,coeff)
%
% Evaluate a polynomial at the points given in x.  The
% coefficients are to be given in coeff.  The constant term
% in the polynomial is coeff(1).

n = length(coeff)
value = coeff(n)*ones(size(x));
for i = n-1:-1:1
  value = coeff(i) + x.*value;
end
```

在 1.1 节末尾给出这个程序更一般的版本. 多项式以嵌套形式求值是 1.3 节研究内容. 在名称为 value 的变量中给出这个函数的输出. 在这个程序中利用记号%表示在该行上记号%后面的有关内容是注释. 作为 polyeval 的例子, 设

$$\text{cf} = [\ 1\quad 2\quad 3\ ];$$

$$\text{t} = \left[\ 3\quad 4\ \right];$$

则指令

$$\texttt{v} = \texttt{polyeval(t, cf);}$$

将使 v 成为数组 [34, 57], 它是多项式 $1+2x+3x^2$ 在 t 的两个分量处的值. 本书中给出了许多函数, 它们能帮助读者更好地学习 MATLAB.

有许多网站帮助你学习 MATLAB, 你可以利用搜索引擎 (例如 Google) 找到许多网站. 只要输入 "Matlab tutorial", 就会发现许多这样的网站. 此外, 有许多优秀的 MATLAB 教材, 一些教材是综述性的, 另一些教材注重于 MATLAB 的某些专题. 综述性的教材我们推荐 Higham and Higham 的 [18]. 关于 MATLAB 图解计算法很完整的介绍, 可参看 Marchand [24].

附录E　二进制数系

大多数计算机使用二进制数系或者它的某种变形而非十进制来表示数和计算. 在本附录中, 我们将介绍二进制以及它与十进制之间的相互转换.

在十进制中, 诸如 342.105 这样的数指的是

$$3 \times 10^2 + 4 \times 10^1 + 2 \times 10^0 + 1 \times 10^{-1} + 0 \times 10^{-2} + 5 \times 10^{-3}$$

一个十进制数可以表为多个 10 的整数次幂的倍数之和. 十进制中有 10 个基本数字, 分别为 $0, 1, \cdots, 9$. 我们把 10 称为十进制的基.

二进制把所有的数字表为多个 2 的整数次幂的倍数之和. 二进制中仅有 2 个数字, 0 和 1; 2 是它的基; 数字 0, 1 叫做*二进制数字*(binary digits), 简称为*位*(bits). 例如, 二进制数 1101.11 的十进制数值是

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \tag{E.1}$$

为了能够清晰地讨论用不同的基表示的数字, 我们将用括号把数字括起来, 再用基做下标. 例如:

$$(1101.11)_2 = (13.75)_{10}$$

如果没有特别说明, 没加括号的数就为十进制数.

为了将一般的二进制数转换成与之等价的十进制数, 我们如 (E.1) 中那样处理. 为此, 有时就要用到几何级数公式:

$$\sum_{i=0}^{n} r^i = \frac{1 - r^{n+1}}{1 - r}, \ r \neq 1 \tag{E.2}$$

或者是它的变形

$$\sum_{i=1}^{n} r^i = \frac{r - r^{n+1}}{1 - r}, \ r \neq 1 \tag{E.3}$$

在式 (E.2) 中, 让 $n \to \infty$, 就得到公式

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1 - r}, \ |r| < 1 \tag{E.4}$$

这个公式在后面将被用到.

例 E.1 考虑在二进制小数点左边的 n 个连续的 1 排列而成的数 $(111\cdots11)_2$. 它与十进制数

$$2^{n-1}+2^{n-2}+\cdots+2^1+2^0=2^n-1$$

是等值的. 因此

$$(\underbrace{11\cdots11}_{n})_2=(2^n-1)_{10} \tag{E.5}$$

类似地,

$$(0.\underbrace{11\cdots1}_{n})_2=2^{-1}+2^{-2}+\cdots+2^{-n}=\frac{2^{-1}-2^{-(n+1)}}{1-2^{-1}}=1-2^{-n}$$

这里我们运用了公式 (E.3), 其中 $r=2^{-1}$. ■

例 E.2 将无限循环的二进制小数

$$x=(0.101\ 010\ 101\ 010\ 1\cdots)_2 \tag{E.6}$$

转换成与之等值的十进制数. 将其转化为 2 的幂, 得到

$$x=2^{-1}+2^{-3}+2^{-5}+2^{-7}+\cdots=\frac{1}{2}\left[1+\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\cdots\right]$$

方括号内的量是一个无穷几何级数, 由公式 (E.4) 可得

$$x=\frac{1}{2}\times\frac{1}{1-\frac{1}{4}}=\frac{1}{2}\times\frac{4}{3}=\frac{2}{3}$$

写成十进制小数为

$$x=(0.666\ 66\cdots)_{10}$$ ■

算术运算的规则在二进制和十进制中是一样的, 主要的差异就是在二进制中只能用很少的数字. 例如, 如下的加法及乘法计算:

$$\begin{array}{rcccccc} & & 1 & 1 & 1 & 1 & 0\\ + & & & 1 & 1 & 0 & 1\\ \hline 1 & 0 & 1 & 0 & 1 & 1 \end{array} \qquad \begin{array}{rrrrrr} & & & 1 & 1 & 1\\ & & \times & 1 & 1 & 0\\ \hline & & & 0 & 0 & 0\\ & & 1 & 1 & 1 & \\ & 1 & 1 & 1 & & \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

表 E-1 和表 E-2 包含了对应于十进制数的 1, 2, 3, 4, 5 的二进制数的加法表和乘法表.

表 E-1　二进制加法

+	1	10	11	100	101
1	10	11	100	101	110
10	11	100	101	110	111
11	100	101	110	111	1 000
100	101	110	111	1 000	1 001
101	110	111	1 000	1 001	1 010

表 E-2　二进制乘法

×	1	10	11	100	101
1	1	10	11	100	101
10	10	100	110	1 000	1 010
11	11	110	1 001	1 100	1 111
100	100	1 000	1 100	10 000	10 100
101	101	1 010	1 111	10 100	11 001

E.1　从十进制到二进制的转换

我们将给出把十进制整数及小数转换成二进制整数及小数的方法. 这些方法对于笔算来说是很便捷的.

假设 x 是一个十进制整数, 我们要求系数 $a_0, a_1, \cdots, a_n$ 都为 0 或 1, $a_n = 1$, 使得

$$a_n 2^n + a_{n-1} 2^{n-2} + \cdots + a_1 2^1 + a_0 2^0 = x \tag{E.7}$$

二进制整数就是

$$(a_n a_{n-1} \cdots a_0)_2 = (x)_{10} \tag{E.8}$$

为了求出系数, 首先用 2 去除 x, 然后用 x_1 表示商, 余数则为 a_0. 接着再用 2 去除 x_1, 用 x_2 来表示商, 余数为 a_1. 继续这种过程, 依次求出 $a_2, a_3, a_4, \cdots, a_n$.

例 E.3　下面就是上述方法的简略形式, 这种形式便于笔算. 将 $(19)_{10}$ 转换成二进制数.

$$\begin{array}{llllllr}
\lfloor 2\overline{)19}\rfloor = 9 & = x_1 & & & & & a_0 = 1 \\
& \lfloor 2\overline{)9}\rfloor = 4 & = x_2 & & & & a_1 = 1 \\
& & \lfloor 2\overline{)4}\rfloor = 2 & = x_3 & & & a_2 = 0 \\
& & & \lfloor 2\overline{)2}\rfloor = 1 & = x_4 & & a_3 = 0 \\
& & & & \lfloor 2\overline{)1}\rfloor = 0 & = x_5 & a_4 = 1
\end{array}$$

在这里, 记号 $\lfloor b \rfloor$ 表示不超过 b 的最大整数. $2\overline{)n}$ 表示 n 除以 2 所得的商. 通过上面的计算可得 $(19)_{10} = (10011)_2$. ■

现在假设 x 是一个小于 1.0 的正的十进制小数, 要求全都为 0 或 1 的系数 $a_1, a_2, a_3, \cdots$, 使得

$$a_1 2^{-1} + a_2 2^{-2} + a_3 2^{-3} + \cdots = x \tag{E.9}$$

则这个二进制小数为

$$(0.a_1 a_2 a_3 \cdots)_2 = (x)_{10} \tag{E.10}$$

为了求出系数, 首先记 $x = x_1$, 用 2 乘以 x_1, 并记 $x_2 = \text{Frac}(2x_1)$, 它表示 $2x_1$ 的小数部分. 整数部分 $\text{Int}(2x_1)$ 等于 a_1. 重复此过程, 用 2 乘以 x_2, 设 $x_3 = \text{Frac}(2x_2)$, $a_2 = \text{Int}(2x_2)$. 依次求出 $a_3, a_4, \cdots$

例 E.4 求 5.578 125 的二进制形式. 把这个数拆成整数部分与小数部分, 如前例中那样, 得

$$(5)_{10} = (101)_2$$

对小数部分, 令 $x_1 = x = 0.578\ 125$, 运用上述算法, 得

$$\begin{array}{lll}
2x_1 = 1.156\ 25 & x_2 = 0.156\ 25 & a_1 = 1 \\
2x_2 = 0.312\ 5 & x_3 = 0.312\ 5 & a_2 = 0 \\
2x_3 = 0.625 & x_4 = 0.625 & a_3 = 0 \\
2x_4 = 1.25 & x_5 = 0.25 & a_4 = 1 \\
2x_5 = 0.5 & x_6 = 0.5 & a_5 = 0 \\
2x_6 = 1.0 & x_7 = 0 & a_6 = 1
\end{array}$$

因此, 相应的二进制数为 0.100 101, 结合刚才的结果, 得到

$$(5.578\ 125)_{10} = (101.100\ 101)_2$$

■

例 E.5 将十进制小数 0.1 转换成等价的二进制数. 通过上面的方法, 就可得到一个无限循环的二进制小数

$$(0.1)_{10} = (0.000\ 110\ 011\ 001\ 10 \cdots)_2 \tag{E.11}$$

一个数拥有有限的二进制小数形式当且仅当它可以表示成有限个 2 的负整数幂之和. 十进数 0.1 就不能够表成这样的有限和. 换句话说, 这个看上去简单的数 0.1 不能够在电脑上精确地表示出来. 这就意味着含有像 0.1 这样简单数字 (更不要说那些更复杂的数了) 的算术运算不能用计算机精确计算. ■

E.2 十六进制数系

十六进制的基是 16, 通常用

$$0, 1, \cdots, 9, A, B, C, D, E, F$$

表示数字. 故 $(F)_{16} = (15)_{10}$. 如前所述, 一个十六进制的数可表为多个 16 的整数幂的倍数之和. 例如,

$$\begin{aligned}(AC.17)_{16} &= (10 \times 16^1 + 12 \times 16^0 + 1 \times 16^{-1} + 7 \times 16^{-2})_{10} \\ &= (172.089\ 843\ 75)_{10}\end{aligned}$$

类似于上面二进制数中所述的那样, 可用这个算法将十进制的数转换为十六进制的数, 参见习题 7.

二进制与十六进制之间有着很密切的联系, 而且两者之间的相互转换也是相当简单的. 从十六进制到二进制, 只需将每个十六进制数字用它对应的四个二进制数字来替换即可.

例 E.6 将 $(AC.17)_{16}$ 转换成二进制形式. 因为

$$(\mathrm{A})_{16} = (1\ 010)_2, \quad \cdots, \quad (7)_{16} = (0\ 111)_2$$

所以

$$(AC.17)_{16} = (10\ 101\ 100.000\ 101\ 11)_2$$ ■

要将 x 从二进制数转换成十六进制数, 从二进制数的小数点的左右两边开始, 可把 x 分割为 4 位一组的数字组, 然后把它们转换为相应的十六进制的数.

例 E.7 将 $x = (110\ 101.110\ 01)_2$ 转换成十六进制数. 则有

$$(00\ 110\ 101.110\ 010\ 00)_2 = (35.C8)_{16}$$ ■

例 E.8 将 (E.6) 中的二进制小数转换成十六进制数. 我们有

$$\begin{aligned}x &= (0.101\ 010\ 101\ 010\ 101\ 0\cdots)_2 \\ &= (0.AAA\cdots)_{16}\end{aligned}$$

正如 (E.6) 后面所述, 它与分数 2/3 等值. ■

许多大型 IBM 计算机上使用十六进制数字, 而其他类型的计算机上主要使用二进制数字.

习题

1. 将下列二进制数转换成十进制形式:

(a) 110 011 **(b)** 101.110 1

(c) 0.100 000 1 **(d)** 1 010 101 010 101

2. 将表 E-1 和表 E-2 都增加一行和一列.

3. 将下列十进制数转换成二进制形式:

(a) 366 **(b)** 4.25 **(c)** 1/6 **(d)** π **(e)** $\sqrt{2}$

4. 求对应于十进制数 0.2, 0.4, 0.8, 1.6 的二进制表达式.

提示: 用 (E.11).

5. 求与无限循环的二进制小数 $x = (0.110\ 110\ 110\cdots)_2$ 所对应的有理数或小数.

提示: 将 x 写为

$$\begin{aligned} x &= 2^{-1} + 2^{-2} + 2^{-4} + 2^{-5} + 2^{-7} + 2^{-8} + \cdots \\ &= [2^{-1} + 2^{-4} + 2^{-7} + 2^{-10} + \cdots] + [2^{-2} + 2^{-5} + 2^{-8} + \cdots] \end{aligned}$$

用 (E.6) 中使用的方法来处理方括号内的每个级数.

6. 将下列十六进制数转换成十进制数及二进制数.

(a) $1F.C$ **(b)** $FF\cdots F$ 重复 n 次 **(c)** 11.1 **(d)** $0.CCCC\cdots$

7. 把十进制转换成二进制的算法推广到把十进制转换成十六进制. 对于整数的转换, 用 16 而不再是用 2 去做除法; 对于小于 1 的十进制小数, 则乘以 16 而不是乘以 2. 将下列十进制数转换成十六进制数.

(a) 161 **(b)** 0.335 937 5 **(c)** 0.1 **(d)** 1.9

8. 设 $0 \leqslant x \leqslant 1$, 将 x 写成二进制形式

$$x = (0.a_1a_2a_3\cdots)_2$$

根据 x 在区间 $[0, 1]$ 内的位置, 给出系数 $a_1, a_2, a_3, \cdots$ 的几何意义.

9. 比较 $(1.000\cdots)_2$ 和 $(0.1111\cdots)_2$ 的大小, 并说明其理由.

部分习题答案

第 1 章

1.1 节

2(a). $p_1(x) = 1 + \frac{1}{2}(x-1)$

$p_2(x) = 1 + \frac{1}{2}(x-1) - \frac{1}{8}(x-1)^2$

3(c). $p_n(x) = 1 + \binom{1/2}{1} x + \binom{1/2}{1} x^2 + \cdots + \binom{1/2}{n} x^n$

9(a). $p_4(x) = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \frac{1}{24}x^4$

11. 对接近于 0 的 x 值, $g(x) \approx 1 + \frac{1}{2}x + \frac{1}{6}x^2$.

13(a). $q(x) = f(a) + [(x-a)/(b-a)][f(b) - f(a)]$

14. $b_0 = f(0)$, $b_1 = f'(0)$; b_2 与 b_3 的值通过解线性方程组

$b_2 + b_3 = f(1) - b_0 - b_1$

$2b_2 + 3b_3 = f'(1) - b_1$

得到.

1.2 节

1. $e^x - p_3(x) = \frac{x^4}{4!}e^c$, c 在 0 和 x 之间

$\max\limits_{-1\leqslant x\leqslant 1} |e^x - p_3(x)| \leqslant (1)^4 \frac{e^1}{24} \doteq 0.113\,3$

从表 1-1 知, $e^1 - p_3(1) \doteq 0.051\,61$.

4(b). 当 $|x| \leqslant 0.245$ 时, $\left|\frac{\sin(x)-x}{x}\right| \leqslant 0.01$.

5. 利用次数 =7.

9(a). $\frac{1}{2}$

14(b). $\tan^{-1}(x) = x - \frac{x^3}{3} + \cdots + \frac{(-1)^n x^{2n+1}}{2n+1} + R_{2n+1}(x)$

$R_{2n+1}(x) = (-1)^{n+1} \int_0^1 \frac{t^{2n+2}}{1+t^2} dt$

19. 当 n 为奇数时, $f(x) = 2\left[x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots + \frac{x^n}{n} + \frac{1}{n+2}\int_0^x \frac{t^{n+2}}{1-t^2} dt\right]$.

1.3 节

4. 利用次数 =8.

6. 利用 (1.35)~(1.38) 式,

$q(x) = 4x^6 + x^5 - x^4 + x^2 + x + 1, \quad b_0 = 0$

$p(x) = (x-1)q(x)$

8. 用 (1.38) 式直接求导形成 $p'(x)$, 然后取 $x = z$.

9. 如果我们假定所有的系数都已算出, 则需要做 7 次乘法.

10. $f(x)=1+z(5+z^2(-1+2z))$, $z=\mathrm{e}^x$.

第 2 章

2.1 节

1(b). $(12)_{10}$ 的 IEEE 双精度十六进制表示为 4 028 000 000 000 000, 所以它的二进制表示为 0 100 000 000 101 000 后随 $4\times 12=48$ 个零.

2. 对 $n=31$ 的有效数字, $M=2^{31}\doteq 2.15\times 10^9$, $-2^{-31}\leqslant\varepsilon\leqslant 2^{-31}$.

2.2 节

1(c). $x_T=2.718\ 281\ 8\cdots$

$x_A=2.714\ 285\ 71\cdots$

$x_T-x_A=0.003\ 996\cdots$

$\mathrm{Re}\,(x_A)\doteq 0.001\ 47$, x_A 有三位有效数字.

3. $\lim\limits_{t\to\infty} N\,(t)=N_c=2N\,(0)$

5(a). $\frac{1-\cos(x)}{x^2}=\frac{2\sin^2(x/2)}{x^2}$

5(d). $\sqrt[3]{1+x}-1=\frac{x}{\left(\sqrt[3]{x+1}\right)^2+\sqrt[3]{x+1}+1}$

6(c). 利用 e^x 和 e^{-x} 的 5 次泰勒级数展开, 则 $\frac{\mathrm{e}^x-\mathrm{e}^{-x}}{2x}\approx 1+\frac{x^2}{3!}+\frac{x^4}{5!}$

利用 e^x 和 e^{-x} 的误差项可以推导误差界.

7. 利用 $\cos u$ 关于 $u=0$ 的 4 次泰勒多项式, 得到

$$\frac{1-\cos(x^2)}{x}=\frac{1}{2}x^3-\frac{1}{24}x^7+\frac{1}{720}x^{11}\cos(c_x),\quad 0\leqslant c_x\leqslant x^2$$

10. 方程的根为: $x_1=20+\sqrt{399}$, $x_2=20-\sqrt{399}$. 为了计算 x_2, 利用 $\sqrt{399}$=19.975, 得到 $x_2=\frac{1}{20+\sqrt{399}}\doteq\frac{1}{39.975}\doteq 0.025\ 016$

19. 对于 IEEE 标准的双精度浮点数, 如果 $|x|<2^{-102.2}\doteq 1.72\times 10^{-31}$, 它将会下溢.

2.3 节

1(a). $2.052\ 65\leqslant x_T+y_T\leqslant 2.053\ 75$

1(b). $\frac{8.472\ 5}{0.064\ 5}\leqslant\frac{x_T}{y_T}\leqslant\frac{8.473\ 5}{0.063\ 5}$

$131.356\leqslant\frac{x_T}{y_T}\leqslant 133.441$

5(a). 利用 (2.4.2), 及 $x_A=1.473$, $|x_T-x_A|\leqslant 0.000\ 5$, 则

$\cos(x_T)-\cos(x_A)=-\sin(c)(x_T-x_A)$

$|\cos(x_T)-\cos(x_A)|\leqslant[\sin(1.473\ 5)](0.000\ 5)<0.000\ 5$

5(c). $x_A=1.471\ 2$, $|x_T-x_A|\leqslant 0.000\ 05$

$|\ln(x_T)-\ln(x_A)|=\frac{1}{c}|x_T-x_A|\leqslant\frac{0.000\ 05}{1.471\ 15}\doteq 0.000\ 034$

6. 如果 x_A 接近于 x_T, $\mathrm{Rel}\,(\sqrt{x_A})\approx\frac{1}{2}\mathrm{Rel}\,(x_A)$.

11. $f(1+10^{-4})\approx(-1)^{n-1}(n-1)!\cdot 10^{-4}$, $f(1)=0$.

当 $n=8$ 时, $f\left(1+10^{-4}\right) \doteq -0.504\,0$.

2.4 节

6. $p-p_n \approx (\varepsilon_2+\cdots+\varepsilon_n)\,p$

第 3 章

3.1 节

1(a). 1.839 287

1(b). 0.424 031

3. r=0.101 34 或者 10.13%

9. $n=32$

15(a). 提示: 首先考虑 $z \geqslant 1$ 的情况.

15(c). 提示: 用 z 的浮点式表示.

3.2 节

2(a). 1.839 286 755

2(c). 0.424 031 039

4. $x_{n+1}=\frac{1}{m}\left[(m-1)x_n+\frac{a}{x_n^{m-1}}\right]$

6.

B	根
1	−0.588 401 776 5
10	−0.326 402 010 1
50	−0.183 291 333 3

3.3 节

8(b). $q_{n+1}=q_n+q_{n-1}$, $q_0=q_1=1$.

3.4 节

2. 利用 $[a,b]=[0,0.5]$. 则当 $a \leqslant x \leqslant b$ 时, $|g'(x)| \leqslant 0.32$.

3. 因为对于所有的 x, 都有 $\mathrm{e}^{-x}>0$, 所以 α 一定是一个正数. 然后考虑 $x>0$, 有 $0<g(x) \leqslant 1$. 这样 $[a,b]=[0,1]$ 满足引理 3.4.1 中的条件 (3.37), 而不满足定理 3.4.2 中的条件 (3.38). 为满足后面的这个条件, 考虑取

$$a=\min_{0 \leqslant x \leqslant 1} g(x) \quad b=\max_{0 \leqslant x \leqslant 1} g(x)$$

5. 利用 $[a,b]=[c-|d|,\ c+|d|]$.

8(b). 收敛, 收敛阶 $=2$.

11. $c=\frac{1}{2}$ 时, 收敛阶为 2.

13(a). 艾特肯误差估计是 -9.175×10^{-4}.

18. 3

3.5 节

4. 对 $\alpha(0)=3$, $\alpha(\varepsilon)\approx 3-15.2\varepsilon=3.03$.

7. $\alpha(\varepsilon)-5\approx 3\ 125\varepsilon$, 是一个病态问题.

8. 根的重数为 4.

第 4 章

4.1 节

1(a). $P_1(x)=2-x$

1(b). $f(x)=\left(\frac{1-2\mathrm{e}}{1-\mathrm{e}}\right)+\left(\frac{1}{1-\mathrm{e}}\right)\mathrm{e}^x$

2(a). $p(x)=3-\cos(\pi x)+2\sin(\pi x)$

7. $-x^2+4x-3$

8(a). $P_2(x)=P_1(x)=x+1$

10. $L_0(x)=\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}$

16. $q(x)=4x^2-4x-1$

19. $f[x_0,\ x_1]=2.363\ 200$, $f'\left(\frac{x_0+x_1}{2}\right)=2.363\ 161$, $f[x_0,x_1,x_2]=1.193\ 495\ 21$

21(a). $p(x)=f(x)-f(x_0)$ 是一个 m 次多项式, 且 $p(x_0)=0$. 因而, $(x-x_0)$ 是 $p(x)$ 的因式, $p(x)/(x-x_0)$ 是 $m-1$ 次多项式.

23(a). $\frac{f'(x_1)-2f[x_0,\ x_1]+f'(x_0)}{(x_1-x_0)^2}$

24.

n	x_n	y_n	$\mathrm{D}y_n$	D^2y_n
0	0.1	0.2		
			0.4	
1	0.2	0.24		1
			0.6	
2	0.3	0.30		

$\mathrm{D}y_n$ 和 D^2y_n 表示一阶和二阶差商. 于是 $P_1(0.15)=0.22$, $P_2(0.15)=0.217\ 5$.

31. 利用 (4.24), 当 $m=n$ 时, 我们得到: $f[x_0,x_1,\cdots,x_n]=\frac{n!}{n!}=1$.

4.2 节

1(a). 1.25×10^{-5}

1(b). 6.42×10^{-8}

3(a). 1.07×10^{-5}

3(b). 1.28×10^{-7}

5(a). $h\leqslant 0.006\ 32$, 比如说, h 取 0.006 或者 0.005.

10. $h^4\mathrm{e}/24$

14. $|\mathrm{e}^x-P_n(x)|\leqslant\frac{\mathrm{e}}{(n+1)n^{n+1}},\quad 0\leqslant x\leqslant 1$

4.3 节

1(c). $s(x)=\begin{cases}x^3+1-x, & 0\leqslant x\leqslant 1\\ (2-x)^3+(4x-3)-(2-x), & 1\leqslant x\leqslant 2\end{cases}$

3(c). 关于 $M_i = s''(x_i)$ 的方程组:

$\frac{1}{3}M_2 + \frac{1}{12}M_3 = 1$

$\frac{1}{12}M_2 + \frac{1}{2}M_3 + \frac{1}{6}M_4 = -\frac{7}{2}$

$\frac{1}{6}M_3 + \frac{2}{3}M_4 = 2$

它的解是: $M_2 = \frac{38}{7}$, $M_3 = -\frac{68}{7}$, $M_4 = \frac{38}{7}$.

7. 关于 $M_i = s''(x_i)$ 的方程组:

$\frac{1}{3}M_1 + \frac{1}{6}M_2 = \frac{1}{2}$

$\frac{1}{6}M_1 + \frac{2}{3}M_2 + \frac{1}{6}M_3 = \frac{1}{3}$

$\frac{1}{6}M_2 + \frac{2}{3}M_3 + \frac{1}{6}M_4 = \frac{1}{12}$

$\frac{1}{6}M_3 + \frac{1}{3}M_4 = \frac{1}{48}$

方程组的解为:

$M_1 = 173/120,\quad M_2 = 7/60,\quad M_3 = 11/120,\quad M_4 = 1/60$

利用 $s(x)$ 的表达式, 我们得到:

$s\left(\frac{3}{2}\right) \doteq 0.652\,604$, 精确值 $f\left(\frac{3}{2}\right) = \frac{2}{3} \doteq 0.666\,667$.

$s\left(\frac{5}{2}\right) \doteq 0.403\,646$, 精确值 $f\left(\frac{5}{2}\right) = 0.4$.

$s\left(\frac{7}{2}\right) \doteq 0.284\,896$, 精确值 $f\left(\frac{7}{2}\right) = \frac{2}{7} \doteq 0.285\,714$.

9. 是.

15. $\beta = 20$, 非自然三次样条.

4.4 节

2(a). $\max\limits_{-1\leqslant x\leqslant 1} |\tan^{-1}(x) - t_1(x)| \doteq 0.215$, $t_3(x) = x - \frac{1}{3}x^3$

2(b). $\max\limits_{-1\leqslant x\leqslant 1} |\tan^{-1}(x) - m_3(x)| \doteq 0.004\,95$

4. 运用三角恒等式, 例如, $\cos(-x) = \cos(x), \cos(x + 2n\pi) = \cos(x)$, n 取任意整数.

5(a). 从 (4.82) 式, $\rho_3(f) \leqslant 0.001\,98$

5(b). $\max\limits_{0\leqslant x\leqslant \pi/2} |\cos(x) - m_3(x)| \doteq 0.001\,37$

8(a). 定义 $\theta = \tan^{-1}(x), x > 0$. 则 $0 < \theta < \pi/2$, $\tan(\theta) = x$. 显然, 利用一个具有角度 θ 的直角三角形即可得到要求的等式: $\tan[(\pi/2) - \theta] = 1/x$, 或者 $(\pi/2) - \theta = \tan^{-1}(1/x)$.

4.5 节

1. $T_5(x) = 16x^5 - 20x^3 + 5x$

3. $x_m = \cos(m\pi/n)$, $m = 0, 1, 2, \cdots, n$.

6. $\left(\frac{1}{2}\right)^{n-1}$

8(c). $S_{n+1}(x) = 2xS_n(x) - S_{n-1}(x), \quad n \geqslant 1$

4.6 节

2. $x_1 = \cos(\pi/4) = 1/\sqrt{2}$, $x_2 = \cos(3\pi/4) = -1/\sqrt{2}$

$P_1(x) = e^{x_1} + (x - x_1)\frac{e^{x_1} - e^{x_2}}{x_1 - x_2} \approx 1.260\,6 + 1.085\,4x$.

3(b). 利用 $T_5(x)$ 的零点, $P_3(x) \approx 0.967\,057\,6x - 0.187\,129\,6x^3$.

$\max_{-1\leqslant x\leqslant 1} |\tan^{-1}(x) - P_3(x)| \doteq 0.005\ 90$

4. 利用 $t = \frac{1}{2}(1+x)$， $-1 \leqslant x \leqslant 1$, 并在 $[-1, 1]$ 上逼近 $f(x) = e^{(1+x)/2}$.

7.

n	极大误差
1	3.72E−2
2	4.37E−3
3	5.72E−4
4	7.94E−5
5	1.14E−5
6	1.69E−6

计算 $p(-x)$ 的值并使它等于 $p(x)$.

4.7 节

1. $\ell_1(x) = (4e - 10) + x(-6e + 18) \doteq 0.873\ 127 + 1.690\ 309x$

3(a). $\ell_3(x) = \ell_4(x) = x\left(-\frac{15}{2\pi} + \frac{315}{2\pi^3}\right) + x^3\left(\frac{35}{2\pi} - \frac{525}{2\pi^3}\right)$
$\doteq 2.692\ 292\ 5x - 2.895\ 604\ 8x^3$

6. $1 = P_0(x)$
$x = P_1(x)$
$x^2 = \frac{1}{3}(2P_2(x) + P_0(x))$
$x^3 = \frac{1}{5}(2P_3(x) + 3P_1(x))$

7. $p(x) = (a_0 + \frac{1}{3}a_2)P_0(x) + a_1P_1(x) + \frac{2}{3}a_2P_2(x)$

第 5 章

5.1 节

1. $T_4(f) \doteq 0.697\ 024$, $I - T_4(f) \doteq -0.003\ 88$
$S_4(f) \doteq 0.693\ 254$, $I - S_4(f) \doteq -0.000\ 107$

2(a).

n	T_n	误差	比率
2	26.516 336	−25.2	
4	3.249 050 5	−1.95	13.0
8	1.624 525 2	−3.22E − 1	6.04
16	1.375 722 5	−7.33E − 2	4.39
32	1.320 311 9	−1.79E − 2	4.09
64	1.306 847 9	−4.45E − 3	4.02
128	1.303 505 7	−1.11E − 3	4.01
256	1.302 671 6	−2.78E − 4	4.00
512	1.302 463 2	−6.95E − 5	4.00

3(a).

n	S_n	误差	比率
2	22.715 077 371	−21.4	
4	−4.506 711 293 0	5.81	−3.69
8	1.083 016 831 5	2.19E − 1	26.5
16	1.292 788 274 5	9.61E − 3	22.8
32	1.301 841 665 3	5.52E − 4	17.4
64	1.302 359 887 9	3.38E − 5	16.3
128	1.302 391 582 8	2.10E − 6	16.1
256	1.302 393 553 1	1.31E − 7	16.0
512	1.302 393 676 1	8.20E − 9	16.0

10(b). $B_4(f) \doteq 0.693\ 174\ 60, I - B_4(f) \doteq -0.000\ 027\ 4$.

13. 2

5.2 节

1(a). $\pi h^2/24$

1(b). $h^2/6$

3(a). $\widetilde{E}_n^T(f) = \frac{-h^2}{12}(\mathrm{e}^\pi - 1)$, $\widetilde{E}_{32}^T(f) \doteq -0.017\ 78$, 真值 $E_{32}^T(f) \doteq -0.017\ 9$.

6(a). $\widetilde{E}_n^s(f) = \frac{13h^4}{1\ 215}$, $h = 2/n$

选择 $h \leqslant 0.031$, 或者 $n \geqslant 65$. 节点数 $= n + 1 \geqslant 66$.

6(d). 对辛普森法, $n \geqslant 396$.

13(b). 利用 $\frac{I_{2n} - I_n}{I_{4n} - I_{2n}} = \frac{(I - I_n) - (I - I_{2n})}{(I - I_{2n}) - (I - I_{4n})}$, 应用 (5.39) 并化简.

14(a). $p = 1.5$

17(a). 从 5.1 节习题 2 (a) 可知, 当 $n = 32$ 时, $I - T_{32} \approx \frac{1}{3}(T_{32} - T_{16}) \doteq -0.018\ 47$. 真值 $I - T_{32} \doteq -0.017\ 9$.

5.3 节

1. $I_3 \doteq 2.350\ 336\ 9$, $I - I_3 \doteq 0.000\ 065\ 5$.

2(a).

n	I_n	误差
2	−19.244 871 326	20.5
3	9.089 728 708 4	−7.79
4	0.922 052 521 5	3.80E − 1
5	1.146 206 344 7	1.56E − 1
6	1.329 809 851 0	−2.74E − 2
7	1.300 343 820 0	2.05E − 3
8	1.302 478 104 1	−8.44E − 5
9	1.302 391 852 9	1.83E − 6
10	1.302 393 688 4	−4.10E − 9

10(a). $\int_0^1 f(x)\ln\left(\frac{1}{x}\right)\mathrm{d}x \approx f\left(\frac{1}{4}\right) \equiv I_1$

10(b). $w_1 + w_2 = 1$

$w_1x_1 + w_2x_2 = \frac{1}{4}$
$w_1x_1^2 + w_2x_2^2 = \frac{1}{9}$
$w_1x_1^3 + w_2x_2^3 = \frac{1}{16}$

10(c). $I_1 \doteq 0.968\ 912$, 误差 $\doteq -0.022\ 8$.

5.4 节

1(a).

h	$\mathrm{D}_h f(0)$	误差估计值	误差	比率
0.1	1.051 709	−5.00E − 2	−5.17E − 2	
0.05	1.025 422	−2.50E − 2	−2.54E − 2	2.03
0.025	1.012 605	−1.25E − 2	−1.26E − 2	2.02
0.012 5	1.006 276	−6.25E − 3	−6.28E − 3	2.01
0.006 25	1.003 132	−3.12E − 3	−3.13E − 3	2.00

3(a).

h	$\mathrm{D}_h f(0)$	误差估计值	误差	比率
0.1	1.001 667 75	−1.67E − 3	−1.67E − 3	
0.05	1.000 416 72	−4.17E − 4	−4.17E − 4	4.00
0.025	1.000 104 17	−1.04E − 4	−1.04E − 4	4.00
0.012 5	1.000 026 04	−2.60E − 5	−2.60E − 5	4.00
0.006 25	1.000 006 51	−6.51E − 6	−6.51E − 6	4.00

5. $f'(x_2) \approx \mathrm{D}_h f(x_2) \equiv (1/12h)\{f(x_0) - 8f(x_1) + 8f(x_3) - f(x_4)\}$
$f'(x_2) - \mathrm{D}_h f(x_2) = (h^4/30) f^{(5)}(c), \quad x_0 \leqslant c \leqslant x_4$

7. $\hat{\mathrm{D}}_h f(x) = 2\mathrm{D}_h f(x) - \mathrm{D}_{2h} f(x)$, 其中 $\mathrm{D}_h f(x)$ 由 (5.76) 式确定.

h	$\mathrm{D}_h f(x)$	$\hat{\mathrm{D}}_h f(x)$	$f'(x) - \hat{\mathrm{D}}_h f(x)$
0.1	−0.542 43		
0.05	−0.521 44	−0.500 45	0.000 45
0.025	−0.510 77	−0.500 10	0.000 10
0.012 5	−0.505 40	−0.500 03	0.000 03

10(b). $f''(x) \approx \frac{1}{h^2}[-f(t+3h) + 4f(t+2h) - 5f(t+h) + 2f(t)]$

12. 记 $\tilde{\mathrm{D}}f(x)$ 为带舍入误差的 $\mathrm{D}_h f(x)$, $\delta = \max\{|\varepsilon_0|, |\varepsilon_1|\}$,
$|f'(x) - \tilde{D}f(x)| \leqslant \frac{h^2}{6}|f'''(c)| + \frac{\delta}{h}, \quad x - h \leqslant c \leqslant x + h$

14. 当 $h = 0.2$, $\mathrm{D}_h^{(2)} f(0.5) \doteq 0.077\ 5$ 时, 由于表中元素舍入的误差部分是 $\frac{4\delta}{h^2} = \frac{4\times(0.000\ 05)}{(0.2)^2}$ $= 0.005$.

第 6 章

6.1 节

1(a). 记 $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, 则 a_0, a_1, a_2, a_3 满足 $a_0 + a_1x_i + a_2x_i^2 + a_3x_i^3 = y_i, \quad i = 0, 1, 2, 3.$

4. $3x_1 + x_2 = 4$

$x_{i-1}+3x_i+x_{i+1}=5,\quad i=2,3,\cdots,n-1$

$x_{n-1}+3x_n=4$

5.
```
sign = 1;
b = zeros (n,1)
A= zeros (n,n);
for i=1: n
   b (i) = sign;
   sign = -sign;
A (i,1: i) =(1/i)*[1: i];
A (1: i-1, i) =A (i, 1: i-1)';
end
```

6.2 节

1(a). $\begin{bmatrix} 5 & 6 \\ -1 & 6 \end{bmatrix}$

1(b). $\begin{bmatrix} a & b+2c \\ 0 & 3c \end{bmatrix}$

1(c). $\boldsymbol{A}=\begin{bmatrix} \frac{7}{9} & -\frac{4}{9} & -\frac{4}{9} \\ -\frac{4}{9} & \frac{1}{9} & -\frac{8}{9} \\ -\frac{4}{9} & -\frac{8}{9} & \frac{1}{9} \end{bmatrix},\quad \boldsymbol{B}=\boldsymbol{I}$

2. 记 $\boldsymbol{A}=[\boldsymbol{A}_{*1},\ \boldsymbol{A}_{*2},\ \boldsymbol{A}_{*3}]$，列矩阵 $\boldsymbol{A}_{*j}$ 表示 $\boldsymbol{A}$ 的第 j 列, 则 $\boldsymbol{AD}=[\lambda_1\boldsymbol{A}_{*1},\lambda_2\boldsymbol{A}_{*2},\lambda_3\boldsymbol{A}_{*3}]$, 这里, 矩阵 $\boldsymbol{A}$ 的第 j 列乘以 λ_j.

5. $\boldsymbol{A}=ww^{\mathrm{T}}$

$\boldsymbol{A}^2=\left(ww^{\mathrm{T}}\right)\left(ww^{\mathrm{T}}\right)=w\left(w^{\mathrm{T}}w\right)w^{\mathrm{T}}=w\,(1)\,w^{\mathrm{T}}=ww^{\mathrm{T}}=\boldsymbol{A}$

8. mnp

17. $a=d,b=c$

28. $[1,1,-2,-2]^{\mathrm{T}}$

6.3 节

1 (a). $\left[\begin{array}{ccc|c} 2 & 1 & -1 & 6 \\ 4 & 0 & -1 & 6 \\ -8 & 2 & 2 & -8 \end{array}\right] \xrightarrow{m_{21}=2,m_{31}=-4} \left[\begin{array}{ccc|c} 2 & 1 & -1 & 6 \\ 0 & -2 & 1 & -6 \\ 0 & 6 & -2 & 16 \end{array}\right]$

$\xrightarrow{m_{32}=-3} \left[\begin{array}{ccc|c} 2 & 1 & -1 & 6 \\ 0 & -2 & 1 & -6 \\ 0 & 0 & 1 & -2 \end{array}\right]$

$x_3=-2,x_2=2,x_1=1$

2. 如果第三个方程是第二个方程的倍数, 则有无限多个解. 解出 x_3 的值, 代入第一个方程, 以确定 x_1 和 x_2 的关系. 如果第三个方程不是第二个方程的倍数, 则方程组不相

容, 此时无解. 为了更好地了解这一特性, 考虑一些具体的方程组.

6(a). $\begin{bmatrix} 0.4 & -0.1 & -0.1 \\ -0.1 & 0.4 & -0.1 \\ -0.1 & -0.1 & 0.4 \end{bmatrix}$

6(d). $\begin{bmatrix} -2/5 & -1/5 & -1/5 & -1/5 \\ -1/5 & -3/5 & -3/5 & -3/5 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & -2 \end{bmatrix}$

7. $MD\,(A \to U) = \frac{1}{2}\,(n+2)\,(n-1)$
$MD\,(b \to g) = n - 1$
$MD\,(g \to x) = \frac{1}{2}n\,(n+1)$

6.4 节

1. $\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1/2 & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & 1/2 \end{bmatrix}.$

2(a). $\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -4 & -3 & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & -2 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$

6(a). $\begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$. 这不是 $\boldsymbol{L}$ 唯一可能的值.

8. $\left[\frac{5}{6},\ -\frac{2}{3},\ \frac{1}{2},\ -\frac{1}{3},\ \frac{1}{6}\right]^{\mathrm{T}}$.

11(a). $MD\,(A \to LU) = 2n - 2$

11(b). $MD\,(f \to x) = 3n - 2$

6.5 节

3(a). $\operatorname{cond}(\boldsymbol{A}) = 289$

5. $\operatorname{cond}(\boldsymbol{A}) = \frac{(1+|c|)^2}{|1-c^2|}$

6. 利用 $\boldsymbol{I} = \boldsymbol{A}\boldsymbol{A}^{-1}$ 以及 (6.88) 中的性质 (P3).

7(b). $\operatorname{cond}(\boldsymbol{A}_n) = n2^{n-1}$

6.6 节

2. 对于雅可比迭代, 比率接近于 0.406 9, 并且 $\|\boldsymbol{M}\| = 1/2$.
对于高斯–赛德尔迭代, 比率不收敛; 但是 $\|\boldsymbol{M}\| = 1/3$.

4(a). $\|\boldsymbol{M}\| = 1/2$

4(b). $\|\boldsymbol{M}\| = 1/2$

7. 对于雅可比迭代, 比率在 0 和 1/2 之间反复.

14. $|\varepsilon| \leqslant \sqrt{2}$

第 7 章

7.1 节

1. $f^*(x) \approx 2.081\ 090\ 9x - 1.083\ 727\ 3,\ E \doteq 0.236$

3. $f^*(x) \approx 7.330\ 107\ 9x^2 - 2.215\ 896\ 1x - 0.981\ 325\ 27$

4. $$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$w_i = \int_0^1 x^{i-1} f(x)\mathrm{d}x, \quad i = 1, 2, 3$$

10. $$na + b\sum_{j=1}^{n} \mathrm{e}^{-x_j} = \sum_{j=1}^{n} y_j$$

$$a\sum_{j=1}^{n} \mathrm{e}^{-x_j} + b\sum_{j=1}^{n} \mathrm{e}^{-2x_j} = \sum_{j=1}^{n} y_j \mathrm{e}^{-x_j}$$

7.2 节

1(a). $f(\lambda) = \lambda^2 - 5\lambda - 6,\ \ \lambda_1 = 6,\ \ \lambda_2 = -1$

$\boldsymbol{v}^{(1)} = [2, 5]^{\mathrm{T}}, \boldsymbol{v}^{(2)} = [1, -1]^{\mathrm{T}}$

2. $\boldsymbol{v}^{(2)} = [1, 0, -1]^{\mathrm{T}}, \boldsymbol{v}^{(3)} = [1, 2, 1]^{\mathrm{T}}$

12. $$\boldsymbol{U} = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \\ -1/\sqrt{3} & 0 & 2/\sqrt{6} \\ 1/\sqrt{3} & -1/\sqrt{2} & 1/\sqrt{6} \end{bmatrix}$$

14. 在这两种情况下，$f(\lambda) = \lambda^2 - 2\lambda + 1 = (\lambda - 1)^2$.

17(b). $\lambda_1 = -12, \lambda_2 = 9, \lambda_3 = 3, \boldsymbol{v}^{(1)} = [1, -1, 1]^{\mathrm{T}}$

17(d). $\lambda_1 = 3, \lambda_2 = 2, \lambda_3 = 1, \boldsymbol{v}^{(1)} = [-4, 3, 1]^{\mathrm{T}}$

17(f). $\lambda_1 = 15, \lambda_2 = 5, \lambda_3 = 5, \lambda_4 = -1, \boldsymbol{v}^{(1)} = [1, 1, 1, 1]^{\mathrm{T}}$

7.3 节

1. 另一个根是 $\alpha \doteq (2.137\ 216\ 736\ 963\ 6,\ 1.052\ 651\ 962\ 811\ 4)$.

2(a). 四个精确的根是 $\left(\pm\sqrt{2.5}, \pm\sqrt{1.5}\right)$ 和 $\left(\pm\sqrt{2.5}, \mp\sqrt{1.5}\right)$.

2(b). 其中的一个根是 $(1.770\ 168\ 921\ 751\ 3,\ 0.465\ 430\ 442\ 833\ 8)$.

第 8 章

8.1 节

1(a). $Y(x) = \sin(x) + \mathrm{e}^{-x}$.

1(b). $Y(x) = x/(x - 0.5)$

3(a). 通解 $Y(x) = c\mathrm{e}^{\lambda x} + \frac{1}{\lambda}\left[\mathrm{e}^{\lambda x} - 1\right]$.

$Y(x) = \left[1 + \frac{1}{\lambda}\right] e^{\lambda x} - \frac{1}{\lambda}$

4(a). $Y = \sqrt{x^2 - 4}$

5(a). $\frac{\partial f}{\partial z} = -1$, 方程良态.

8.2 节

1 (a).

x	$Y(x)$	$Y(x) - y_{0.2}(x)$	$Y(x) - y_{0.1}(x)$	$Y(x) - y_{0.05}(x)$
2	1.107 15	−3.32E−2	−1.63E−2	−8.11E−3
4	1.325 82	−1.67E−2	−8.34E−3	−4.17E−3
6	1.405 65	−9.70E−3	−4.86E−3	−2.44E−3
8	1.446 44	−6.38E−3	−3.20E−3	−1.60E−3
10	1.471 13	−4.54E−3	−2.28E−3	−1.14E−3

2(c).

x	$Y(x)$	$Y(x) - y_{0.5}(x)$	$Y(x) - y_{0.0625}(x)$
2	0.493 15	2.23E−1	−4.44E−3
4	−1.410 45	1.34E+0	8.67E−3
6	0.680 75	6.39E+0	−2.78E−3
8	0.843 86	3.25E+1	−6.36E−3
10	−1.383 09	1.65E+2	8.07E−3

8.3 节

3. $K = \sup_{0 \leqslant x \leqslant b} (1/(x+1)) = 1$, 对任意的 $b \geqslant 0$.

$\max_{0 \leqslant x \leqslant b} |Y''(x)| = \max_{0 \leqslant x \leqslant b} (2x/(x+1)) = 2b/(b+1)$

$|Y(b) - y_h(b)| \leqslant h \left[\frac{e^b - 1}{2}\right] \left[\frac{2b}{b+1}\right]$

$h = 0.05$ 时的数值结果:

b	$Y(b) - y_h(b)$	误差界
1	0.018 7	0.043 0
2	0.063 4	0.213
3	0.125	0.716
4	0.199	2.14

5(b). 利用 $K = 2 \max_{0 \leqslant x \leqslant b} |4Y(x)| = 4$, 对 $b \geqslant 1$.

$\max_{0 \leqslant x \leqslant b} |Y''(x)| = \max_{0 \leqslant x \leqslant b} \left|\frac{-6x + 2x^3}{(1+x^2)^3}\right| \doteq 1.46$, 对 $b \geqslant 0.41$.

7. 求解 $D'(x) = \lambda D(x) - \frac{1}{2}\sin(x), D(0) = 0$, 得到 $D(x) = \{[\lambda \sin(x) + \cos(x)] - e^{\lambda x}\} / [2(\lambda^2 + 1)]$ 且 $Y(x_n) - y_h(x_n) \approx D(x_n)h$.

当 $\lambda = -1$ 和 $h = 0.1$ 时, 结果如下:

x	$D(x)h$	$Y(x)-y_h(x)$
1	−0.016 7	−0.016 9
2	−0.036 5	−0.037 2
3	−0.029 5	−0.029 9
4	0.002 12	0.002 58
5	0.030 9	0.031 9
6	0.030 9	0.031 6

10(a). 对 $2h=0.1$, $h=0.5$ 的情况,

x	$Y(x)-y_h(x)$	$y_h(x)-y_{2h}(x)$	$\widetilde{y}_h(x)$	$Y(x)-\widetilde{y}_h(x)$
2	−8.11E−3	−8.22E−3	1.107 029 6	1.19E−4
4	−4.17E−3	−4.16E−3	1.325 816 5	1.12E−6
6	2.44E−3	−2.43E−3	1.405 654 9	−7.28E−6
8	−1.60E−3	−1.60E−3	1.446 447 2	−5.84E−6
10	−1.14E−3	−1.14E−3	1.471 131 7	−4.05E−6

8.4 节

7. 我们给出 $\lambda=-50$ 时由向后欧拉方法得到的值. 对于同一个 h 比较 $\lambda=-1$ 和 $\lambda=-10$ 时的这些值. 注意到当 h 从 0.1 降低到 0.01 时, 误差大约减小到 1/10; 这个结果与方法具有一阶收敛速度是相容的.

	误差 $Y(x)-y_h(x)$		
x	$h=0.5$	$h=0.1$	$h=0.01$
2	1.01E−3	1.70E−4	1.64E−5
4	1.59E−4	2.88E−5	2.81E−6
6	4.82E−5	9.00E−6	8.87E−7
8	2.03E−5	3.86E−6	3.82E−7
10	1.04E−5	1.99E−6	1.97E−7

8.5 节

2(a). 下面是 $h=0.05$ 和 $2h=0.1$ 的一些结果, 其中包括由 (8.76) 给出的理查森误差估计.

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$
2	2.05E−4	5.12E−5	4.0	5.12E−5
4	1.75E−4	4.28E−5	4.1	4.40E−5
6	1.01E−4	2.48E−5	4.1	2.55E−5
8	6.40E−5	1.57E−5	4.1	1.61E−5
10	4.37E−5	1.07E−5	4.1	1.10E−5

5.

x	$Y(x)-y_h(x)$	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$	$\widetilde{y}_h(x)$	$Y(x)-\widetilde{y}_h(x)$
2	2.31E−4	2.31E−4	0.493 151 3	−7.24E−7
4	2.91E−4	3.07E−4	−1.410 430 0	−1.61E−5
6	−4.08E−4	−4.19E−4	0.680 743 2	1.16E−5
8	5.68E−5	5.08E−5	0.843 852 2	6.05E−6
10	3.62E−4	3.78E−4	−1.383 076 0	−1.67E−5

6. 当 $\gamma_2 = 1$ 时, 龙格–库塔方法是

$$y_{n+1} = y_n + hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n)\right)$$

7. 利用习题 6 中 $\gamma_2 = 1$ 时的公式, 我们得到 $h = 0.05$, $2h = 0.1$ 时的误差结果, 其中包括理查森误差估计.

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$
2	1.46E−3	3.53E−4	4.1	3.69E−4
4	−6.68E−4	−1.67E−4	4.0	−1.67E−4
6	−8.31E−4	−1.97E−4	4.2	−2.11E−4
8	1.37E−3	3.33E−4	4.1	3.46E−4
10	−3.08E−4	−8.01E−5	3.8	−7.60E−5

8(a). 当 $h = 0.05$ 和 $2h = 0.1$ 时,

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$
2	6.46E−4	1.56E−4	4.1	1.63E−4
4	2.51E−4	6.10E−5	4.1	6.34E−5
6	1.26E−4	3.07E−5	4.1	3.18E−5
8	7.49E−5	1.82E−5	4.1	1.89E−5
10	4.94E−5	1.20E−5	4.1	1.25E−5

9. 下面是 $h = 0.125$ 和 $2h = 0.25$ 时的结果, 连同从习题 4 得出的对 $y_h(x)$ 的理查森误差估计.

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{15}[y_h(x)-y_{2h}(x)]$
2	3.02E−5	1.69E−6	18	1.90E−6
4	−3.64E−5	−2.20E−6	17	−2.28E−6
6	−1.44E−6	3.41E−8	−42*	−9.85E−8
8	3.74E−5	2.16E−6	17	2.35E−6
10	−2.97E−5	−1.83E−6	16	−1.86E−6

* 因为比率 < 0, 理查森外推将会不准确.

13(a). 真解是 $v(t) = -98\left[1 - e^{-0.1t}\right]$, $t \geqslant 0$.

8.6 节

1(a). 当 $h=0.05$, $2h=0.1$ 时,

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$
2	5.10E−4	1.28E−4	4.0	1.27E−4
4	4.37E−4	1.07E−4	4.1	1.10E−4
6	2.54E−4	6.23E−5	4.1	6.40E−5
8	1.61E−4	3.93E−5	4.1	4.04E−5
10	1.09E−4	2.68E−5	4.1	2.76E−5

2(a). 当 $h=0.05$, $2h=0.1$ 时. 理查森误差公式来自 8.5 节习题 4. 收敛阶是 $p=3$. y_1 和 y_2 利用 $\gamma_2=1$ 的二阶 RK 方法得到的.

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{7}[y_h(x)-y_{2h}(x)]$
2	1.27E−4	1.49E−5	8.5	1.60E−5
4	5.17E−6	6.53E−7	7.9	6.45E−7
6	−1.55E−6	−1.65E−7	9.4	−1.98E−7
8	−1.72E−6	−1.94E−7	8.9	−2.17E−7
10	−1.36E−6	−1.56E−7	8.7	−1.72E−7

4(a). 当 $h=0.05$, $2h=0.1$ 时,

x	$Y(x)-y_{2h}(x)$	$Y(x)-y_h(x)$	比率	$\frac{1}{3}[y_h(x)-y_{2h}(x)]$
2	−1.47E−4	−3.14E−5	4.7	−3.86E−5
4	−1.08E−4	−2.40E−5	4.5	−2.80E−5
6	−6.11E−5	−1.37E−5	4.5	−1.58E−5
8	−3.81E−5	−8.61E−6	4.4	−9.84E−6
10	−2.58E−5	−5.85E−6	4.4	−6.66E−6

6(a). 下面给出对应于三种步长的误差. 明显不同的误差值是由于方法具有的不同的稳定性区域.

x	$Y(x)-y_{0.1}(x)$	$Y(x)-y_{0.02}(x)$	$Y(x)-y_{0.01}(x)$
2	3.82E+13	1.37E−4	1.09E−6
4	2.06E+30	1.33E−4	−5.54E−8
6	1.11E+47	1.29E−4	−1.05E−6
8	5.98E+63	1.37E−4	9.27E−7
10	3.23E+80	1.34E−4	2.76E−7

8.7 节

1. $Y_1'=Y_1-2Y_2-2e^{-x}+2,\quad Y_1(0)=1$

$Y_2'=2Y_1-Y_2-2e^{-x}+1,\quad Y_2(0)=1$

3(b). 设 $Y_1=Y$, $Y_2=Y'$, 得

$$Y' = \begin{bmatrix} 0 & 1 \\ -13 & -4 \end{bmatrix} Y + \begin{bmatrix} 0 \\ 40\cos(X) \end{bmatrix}, \quad Y(0) = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

7(b). 利用习题 3(b) 的记号. 误差估计是

$$y(x) - y_h(x) \approx y_h(x) - y_{2h}(x)$$

		$Y_1(x) = y(x)$ 中的误差		
x	$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.025$ 时的误差估计
2	−2.13E−2	−5.12E−3	−1.65E−3	−3.47E−3
4	4.03E−2	2.09E−2	1.06E−2	1.03E−2
6	−5.14E−2	−2.54E−2	−1.26E−2	−1.28E−2
8	4.17E−4	−3.74E−4	−3.31E−4	−4.35E−5
10	5.10E−2	2.57E−2	1.29E−2	1.28E−2

8(b). 利用习题 3(b) 的记号. 误差估计是

$$y(x) - y_h(x) \approx \frac{1}{3}\left[y_h(x) - y_{2h}(x)\right]$$

		$Y_1(x) = y(x)$ 中的误差		
x	$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.025$ 时的误差估计
2	8.20E−4	1.74E−4	4.06E−5	4.44E−5
4	−7.97E−3	−2.02E−3	−5.04E−4	−5.06E−4
6	4.77E−3	1.32E−3	3.44E−4	3.27E−4
8	3.95E−3	9.06E−4	2.15E−4	2.30E−4
10	−8.05E−3	−2.08E−3	−5.23E−4	−5.19E−4

9(a). $c_1 = 3,\ c_2 = 2$.

8.8 节

3(a). $Y(x) = \frac{e^{2-x} - e^x}{e^2 - 1}$

3(b). $Y(x) = \frac{e^{2-x} + e^x}{e^2 + 1}$

3(c). $Y(x) = \frac{e^x - e^{2-x}}{e^2 + 1}$

7. 真解是 $Y(x) = xe^x$. 下表列出在某些点处的数值误差 $Y(x) - y_h(x)$.

x	$h = 1/10$	$h = 1/20$	$h = 1/40$
1.2	3.40E−3	8.43E−4	2.11E−4
1.4	6.30E−3	1.57E−3	3.91E−4
1.6	7.71E−3	1.92E−3	4.78E−4
1.8	6.24E−3	1.55E−3	3.87E−4

第 9 章

9.1 节

5. 对应于 $n_x = n_y = 5$ 的数值解是

	$x = 1/5$	$x = 2/5$	$x = 3/5$	$x = 4/5$
$y = 1/5$	1.116 1	1.155 8	1.177 6	1.190 6
$y = 2/5$	1.268 5	1.329 4	1.363 9	1.384 8
$y = 3/5$	1.468 5	1.529 4	1.563 9	1.584 8
$y = 4/5$	1.716 1	1.755 8	1.777 6	1.790 6

7. 最大的数值解误差是

n	最大误差	比率
4	2.176 1E−1	
8	5.834 0E−2	3.73
16	1.484 3E−2	3.93
32	3.708 7E−3	4.00

8. 对于 $2 \leqslant i, j \leqslant n$, 差分方程是

$$u_{i,j} = \frac{1}{4 - ah^2}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right) - \frac{h^2}{4 - ah^2} f_{ij}$$

9.2 节

4. 显式格式的数值解最大误差是

	$n_t = 5$	$n_t = 10$	$n_t = 20$	$n_t = 40$
$n_x = 5$	2.066 0E−3	3.587 7E−3	4.334 8E−3	4.704 9E−3
$n_x = 10$	1.964 6E−3	2.899 6E−4	5.317 8E−4	9.388 6E−4
$n_x = 20$	3.009 0E−3	1.315 3E−3	4.842 8E−4	7.263 0E−5
$n_x = 40$	3.270 7E−3	1.572 1E−3	1.483 9E−2	2.386 3E−3
$n_x = 80$	3.336 5E−3	5.590 5E+0	2.135 6E+11	3.820 6E+26

隐式格式的数值解最大误差是

	$n_t = 5$	$n_t = 10$	$n_t = 20$	$n_t = 40$
$n_x = 5$	7.939 2E−3	6.522 9E−3	5.802 2E−3	5.438 6E−3
$n_x = 10$	4.493 3E−3	2.937 3E−3	2.145 2E−3	1.745 5E−3
$n_x = 20$	3.521 2E−3	1.948 1E−3	1.147 2E−3	7.430 8E−4
$n_x = 40$	3.277 6E−3	1.700 2E−3	8.971 3E−4	4.918 9E−4
$n_x = 80$	3.216 7E−3	1.638 2E−3	8.345 7E−4	4.290 6E−4

9. 我们使用隐式格式, 并由 $n_t = n_x = 4$ 开始. 由于格式在时间上是一阶的, 在空间上是二阶的, 我们连贯地将 n_t 乘以 2 和将 n_x 乘以 4. 数值结果 $(\times 10^{-3})$ 的序列是:

6.155 2, 7.269 5, 7.487 4, 7.536 5, 7.548 5, 7.551 4, 7.552 2.

三个稳定的数字是 7, 5, 5, 序列的最后一个数与 $n_x = 256, n_t = 16\ 384$ 相对应.

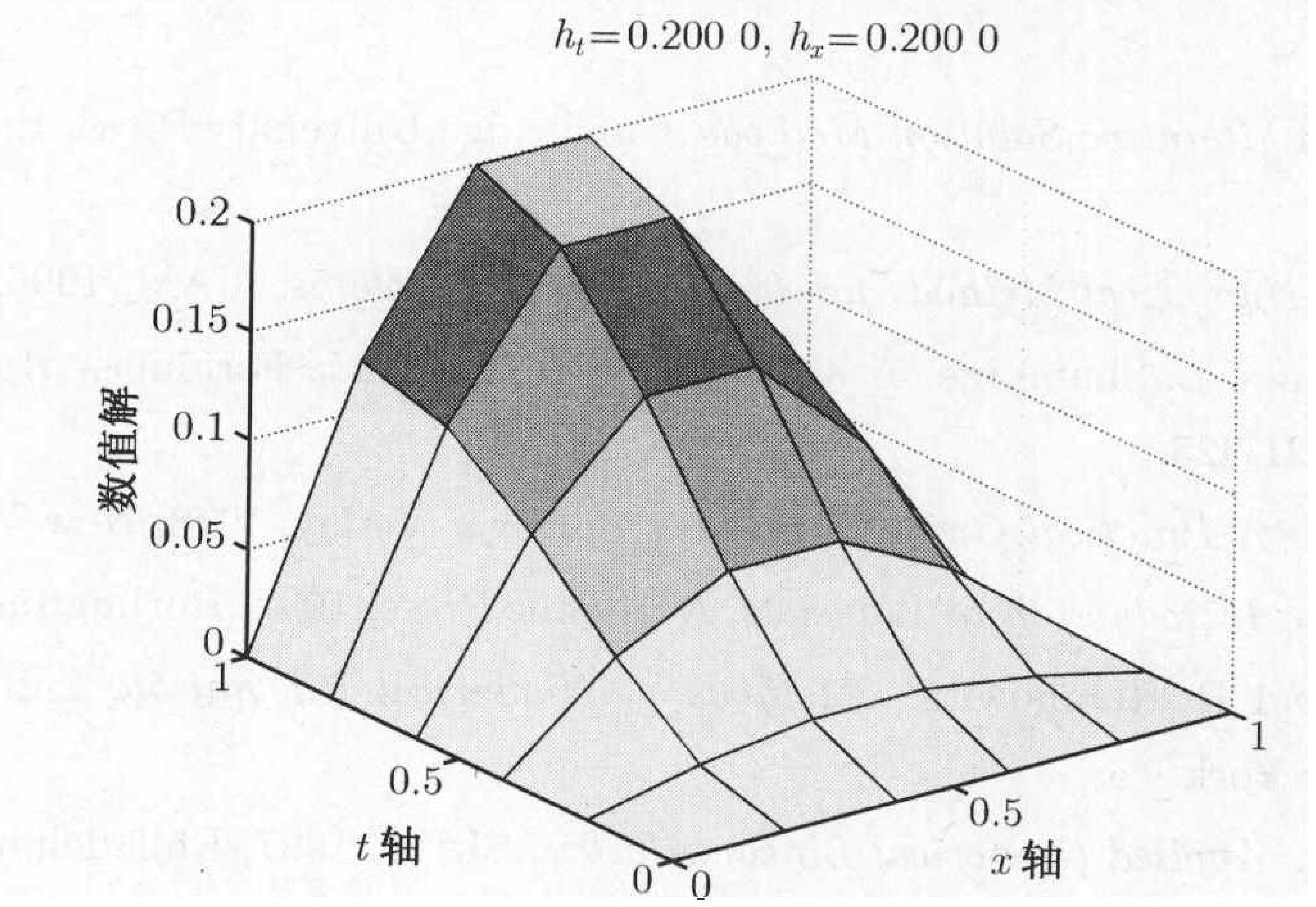
9.3 节

4. 在 $t = 1$ 处的几个数值解的值：

	$x = 1/5$	$x = 2/5$	$x = 3/5$	$x = 4/5$
$n_x = n_t = 5$	−0.442 0	−0.877 6	−1.039 4	−0.666 1
$n_x = n_t = 10$	−0.465 6	−0.946 3	−1.006 8	−0.663 5
$n_x = n_t = 20$	−0.472 1	−0.962 4	−1.005 1	−0.660 8

7(a). 当 $n_t = n_x = 5$ 时, 下表中给出数值解的值 ($t \leqslant 1$), 并在图中指出近似的垂直位移的图形.

	$x = 1/5$	$x = 2/5$	$x = 3/5$	$x = 4/5$
$t = 0.2$	0.011 8	0.019 0	0.019 0	0.011 8
$t = 0.4$	0.042 5	0.068 8	0.068 8	0.042 5
$t = 0.6$	0.080 6	0.130 4	0.130 4	0.080 6
$t = 0.8$	0.111 4	0.180 2	0.180 2	0.111 4
$t = 1$	0.123 1	0.199 2	0.199 2	0.123 1



图： 近似的垂直位移

附录 E

1(b). $(5.812\ 5)_{10}$

3(b). $(100.01)_2$

3(d). $\pi \doteq (11.001\ 001)_2$

4. $(0.8)_{10} = (0.110\ 011\ 001\ 100\cdots)_2$

5. 6/7

参考文献

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*, Dover Press, 1964, New York.

[2] M. Allen III and E. Isaacson. *Numerical Analysis for Applied Science*, John Wiley, 1998, New York.

[3] E. Anderson, Z. Bai. C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson. *LAPACK Users' Guide*, SIAM, 1992, Philadelphia.

[4] U. Ascher, R. Mattheij, and R. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, 1988, Englewood Cliffs, New Jersey.

[5] K. Atkinson. *An Introduction to Numerical Analysis*, 2nd ed., John Wiley, 1989, New York.

[6] O. Axelsson. *Iterative Solution Methods*, Cambridge University Press, 1994, Cambridge, U. K.

[7] A. Bjorck. *Numerical Methods for Least Squares Problems*, SIAM, 1996, Philadelphia.

[8] P. Bogacki and L. Shampine. A 3(2) Pair of Runge-Kutta Formulas. *Appl. Math.. Lett.* **2** (1989), 321-325.

[9] C. de Boor. *A Practical Guide to Splines*, Springer-Verlay, 1978, New York.

[10] C. Chui. *An Introduction to Wavelets*, Academic Press, 1992, Burlington, Masschusetts.

[11] P. Davis and P. Rabinowitz. *Methods of Numerical Integration*, 2nd ed., Academic Press, New York.

[12] J. Demmel. *Applied Numerical Linear Algebra*, SIAM, 1997, Philadelphia.

[13] J. Dongarra, I. Duff, D. Sorensen, and H. van der Vorst. *Numerical Linear Algebra for High-Performance Computers*, SIAM, 1998, Philadelphia.

[14] J. Dormand and P. Prince. A Family of Embedded Runge-Kutta Formulae, *J. Comp. Appl. Math* **6**(1980), 19-26.

[15] W. Gander and J. Hrebicek. *Solving Problems in Scientific Computing Using Maple and Matlab*, 3rd ed., Springer, 1997, New York.

[16] W. Gautschi. *Numerical Analysis: An Introduction*, Birkhäuser, 1997, Boston.

[17] G. Golub and C. Van Loan. *Matrix Computations*, 3rd ed., John Hopkins University Press, 1996, Baltimore.

[18] D. Higham and N. Higham. *Matlab Guide*, SIAM, 2000, Philadelphia.

[19] A. Householder. *The Numerical Treatment of a Single Nonlinear Equation*, McGraw-Hill, 1970, New York.

[20] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, 1996, Cambridge, U. K.

[21] C. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995, Philadelphia.

[22] R. Kress. *Numerical Analysis*, Springer-Verlag, 1998, New York.

[23] A. Krommer and C. Ueberhuber. *Computational Integration*, SIAM, 1998, Philadelphia.

[24] P. Marchand. *Graphics and GUIs with MATLAB*, 2nd ed., CRC Press, 1999, Boca Raton, Florida.

[25] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*, Cambridge University Press, 1994, Cambridge, U. K.

[26] J. Nocedal and S. Wright. *Numerical Optimization*, Springer-Verlag, 1999, New York.

[27] M. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, 2001, Philadelphia.

[28] B. Parlett. *The Symmetric Eigenvalue Problem*, Classics in Applied Mathematics, SIAM, 1998, Philadelphia.

[29] M. Powell. *Approximation Theory and Methods*, Cambridge University Press, 1981, Cambridge, U. K.

[30] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*, Springer-Verlay, 2000, New York.

[31] L. Shampine. *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, 1994, New York.

[32] A. Stroud and D. Secrest. *Gaussian Quadrature Formulas*, Prentice-Hall, 1966, Englewood Cliffs, New Jersey.

[33] L. Trefethen and D. Bau. *Numerical Linear Algebra*, SIAM, 1997, Philadelphia.

索　引